SENIOR DESIGN PROJECT

ECE 186B

LuLi Smart Cultivation: A Socially-Connected IoT Hydroponics Solution

California State University, Fresno
Lyles College of Engineering
Electrical and Computer Engineering Department

Dr. Gregory Kriehn
Professor Roger Moore

BY

Liam Goss
Luigi Santiago-Villa

Instructor: _____

Name Signature Date

Technical Advisor: _____

Name Signature Date

Contents

List of Tables

List of Figures

Abstract

This project addresses food deserts by developing an accessible, internet-connected hydroponics system, enabling urban households to grow nutritious food. Leveraging temperature, humidity, light intensity, and water pH sensors, alongside UV lighting and a microcontroller, the system facilitates real-time plant health monitoring. A unique feature is its community-driven web application, allowing users to exchange growing tips and presets, significantly lowering the entry barrier for novices. Designed with a focus on food desert challenges and community engagement, the project aims to make home-grown crops viable for everyone. Over nine months and with a $500 budget, this initiative will produce a functional prototype that demonstrates efficient crop cultivation, highlighting a technology-driven, socially inclusive approach to mitigate food insecurity in urban areas.

Overview

The hydroponics system has been successfully developed centered around user-friendliness, comprehensive data analytics, and internet connectivity. Functioning seamlessly as a household appliance, the initiative effectively addresses the crucial need for access to nutritious food in areas identified as food deserts. The automated system comprises multiple sensors for monitoring temperature, humidity, light intensity, and water pH, alongside a UV lighting system, water reservoir, water pump, and a microcontroller. These integrated components provide end-users with real-time insights into plant health and nutrient metrics.

Additionally, a web application has been developed to facilitate user interaction, enabling the sharing of insights, posing queries, and exchanging settings presets. Serving as the central nexus for accessing sensor data and engaging with community-driven inputs, this platform significantly enhances user experience. Notably, the solution stands out from existing market offerings because of its meticulous design tailored to address the unique challenges of food deserts and foster reliance on local communities through social networking.

The project, spanning two college semesters and totaling nine months, adhered to a budget allocation of approximately $500. Major milestones were achieved through meticulous planning and execution, encompassing design and implementation phases, further subdivided into research, component sourcing, component testing, and final integration. The resulting fully functional prototype, constructed from Oriented Strand Board (OSB), incorporates the aforementioned sensors, a microcontroller, lighting apparatus, and water management controls. The completion of a fully operational prototype was accomplished early in the project timeline, allowing for the cultivation of basic crops to vividly demonstrate the efficacy and functionality of the design.

1 Design, Methodology, and Implementation

The hydroponics system was designed to address the challenges of food deserts by providing urban households with a sustainable and accessible means of growing fresh produce. The system's design methodology focused on integrating IoT capabilities, real-time monitoring, and community engagement features to enhance user experience and foster a sense of community among users. The implementation process involved sourcing components, designing the circuit, testing individual components, and integrating them into a functional system. The resulting hydroponics system comprises multiple sensors, a microcontroller, a water pump, and a UV lighting system, all working together to create an automated growing environment. The system's web application serves as a central hub for users to monitor plant health metrics, share insights, and exchange growing tips, enhancing the overall user experience. The project's successful completion demonstrates the feasibility of using technology to address food insecurity in urban areas and promote community-driven solutions to social challenges.

1.1 Circuit Design

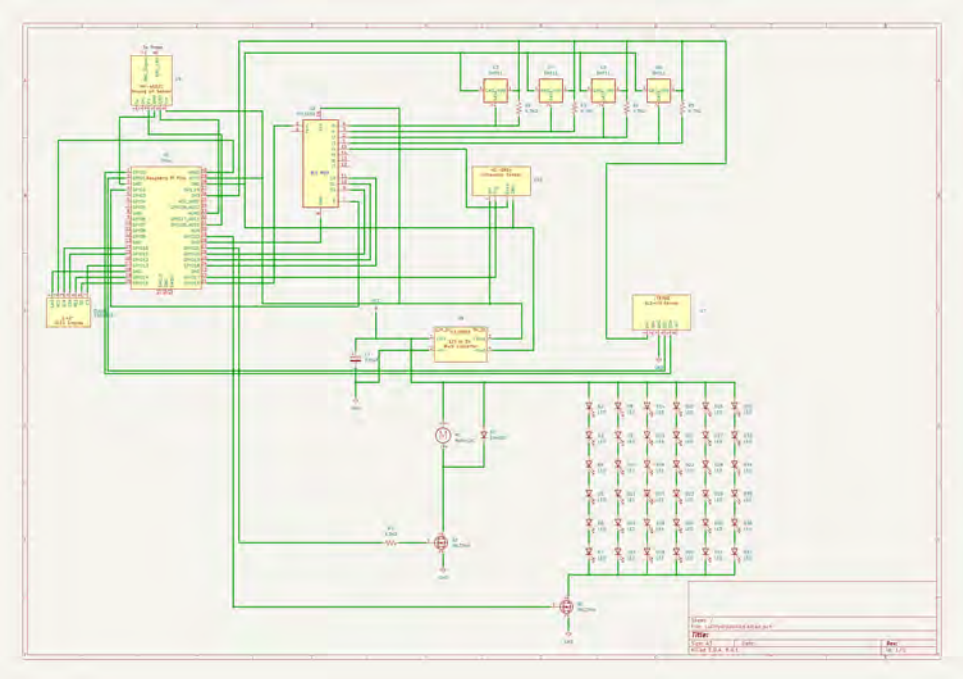


Figure 1: Circuit Schematic Overview

The hydroponics system's circuit design was specially crafted to ensure the seamless integration of its diverse components. These include multiple sensors, a microcontroller, a water pump, and a UV lighting system, each fulfilling a distinct role within the automated hydroponic framework. The design facilitates real-time monitoring of environmental conditions, nutrient levels, and plant health metrics, empowering users to optimize growth conditions and ensure robust crop development.

Comprehensive sensor integration encompasses temperature, humidity, light intensity, and water pH, enabling thorough data collection crucial for plant growth assessment. The microcontroller acts as the system's core, gathering sensor data, managing component operations, and supporting user interaction through a web interface. Essential components like the water pump and UV lighting system play vital roles in sustaining optimal growth conditions by ensuring adequate hydration and nutrient distribution.

The circuit design was meticulously planned to accommodate diverse power demands, establish effective communication protocols between sensors and the microcontroller, and enable seamless data exchange for real-time monitoring and control. This systematic approach resulted in a fully operational automated growing environment. The resulting circuit configuration forms a sturdy foundation for monitoring and managing key parameters essential for successful plant cultivation, allowing users to foster healthy crops with minimal manual intervention.

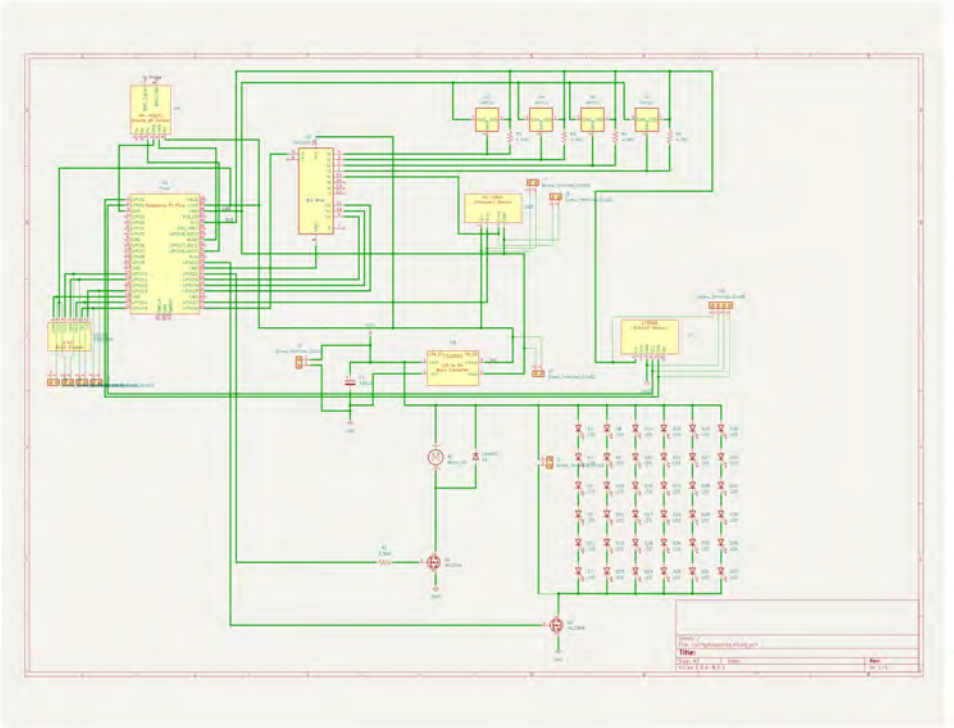


Figure 2: Circuit Schematic Designed with Terminal Blocks

In the final month of the project's development, a PCB was designed to house the system's components, providing a compact and organized layout that enhances reliability and ease of maintenance. In order to connect the sensors placed at various distances around the housing, terminal blocks were used to extend the connections. The KiCad PCB designer required that the schematic include these screw terminals to ensure that all nets and wires were connected correctly. The updated schematic (2) was then used to generate the Gerber files for manufacturing the PCB.

1.2 Component Selection

1. **Microcontroller**
   - Raspberry Pi Pico W
2. **Water Pump**
   - CrocSee 12VDC Diaphragm Pump
3. **Power Supply Unit (PSU)**
   - BMOUO 12V AC-to-DC Power Supply
   - FILSHU 10A 250V Power Socket Inlet Switch
4. **Buck Converter**
   - DORHEA C120503 12V to 5V DC Converter
5. **Sensors**
   - Four DHT22 Sensors
   - LTR-390UV-01 Light Sensor
   - HC-SR04 ultrasonic sensor
   - PH-4502C analog pH sensor
6. **Additional Circuit Parts**
   - Flyback diode (1N4001RLGOSCT-ND)
   - 220µF Capacitor (16ZLH220MEFCT16.3X11)
   - Two N-Channel MOSFET (IRLZ34NPBF-ND)
   - 8:1 MUX (SN74LS151N)
   - 2.42" SSD1309 OLED display
   - 60W full spectrum grow lights
   - 18 AWG wire
   - 22 AWG wire

1.3 Circuit Power Supply Design

A 12V30A power supply unit (PSU) was selected to provide the necessary power for the hydroponics system. The PSU was chosen for its high current capacity, ensuring that all system components receive adequate power for optimal performance. The PSU is connected to a FILSHU 10A 250V power socket inlet switch, allowing for easy power management and control. The power supply unit is responsible for delivering the required voltage and current to the system's components, ensuring reliable operation and consistent performance. The PSU's high current capacity enables it to power multiple devices simultaneously, supporting the system's diverse functionalities and ensuring robust operation under varying load conditions.

Connected to the PSU is a DORHEA C120503 12V to 5V DC converter, which steps down the voltage to 5V for powering the Raspberry Pi Pico W microcontroller and various 5V sensors. The DC converter is essential for providing the microcontroller with the appropriate voltage level, ensuring stable operation and preventing damage due to overvoltage. The converter's efficient design and reliable performance make it an ideal choice for powering the microcontroller, enabling seamless integration into the hydroponics system. By converting the 12V input voltage to 5V, the DC converter ensures compatibility with the microcontroller's operating specifications, facilitating reliable communication and control of the system's components.

The inclusion of a 220uF capacitor [11] in the circuit plays a critical role in maintaining the stability and integrity of the power supply, particularly given the dynamic nature of the loads present. This capacitor acts primarily as a decoupling device, providing voltage smoothing to mitigate fluctuations attributable to transient changes in the load, such as those occurring during the activation or deactivation of the motor, or the variable current demands of the LED array. It serves to absorb and release electrical energy, thereby ensuring a consistent voltage supply despite abrupt variations. Furthermore, the capacitor contributes to noise reduction by filtering out electrical interference generated by the motor and the pulsating operation of the LEDs, preserving other sensitive components from potential disruptions. Notably, the buck converter, which is susceptible to input voltage irregularities, benefits from the presence of the capacitor that acts as a buffer against voltage spikes. Moreover, the capacitor enhances the power supply's transient response, promptly supplying current during sudden demand surges, allowing the power supply to adapt effectively to the changing load requirements. This preventive measure ensures both the longevity and performance of the circuit components are optimized, and the risk of malfunctions is minimized.

1.4 LED Control Circuit

In the development of the LED control system, a low-side switching configuration was implemented using the IRLZ34N N-channel MOSFET. This configuration effectively allows the modulation of a 12V LED strip via signals originating from a Raspberry Pi Pico W microcontroller. The circuit is designed such that the source terminal of the IRLZ34N MOSFET is connected directly to the ground. The drain terminal is connected to the ground terminal of the LED strip, facilitating the control of the negative power line of the strip. The gate terminal of the MOSFET is interfaced with GPIO22 on the Raspberry Pi Pico W, enabling digital control of the LED strip's operational state. The positive terminal of the LED strip is connected to a 12V power supply, ensuring the provision of the necessary driving voltage for the LEDs.

The IRLZ34N MOSFET [10] operates based on the principles of semiconductor physics concerning the movement of electrons and the manipulation of charge carriers within the device. As an N-channel MOSFET, the primary charge carriers are electrons, which are more mobile than holes. The MOSFET is constructed with a channel between the source and the drain, made of N-type semiconductor material. When a positive voltage is applied to the gate relative to the source, it creates an electric field that induces a conduction channel in the underlying silicon structure. This channel allows electrons to flow from the source to the drain when the gate voltage exceeds a certain threshold, thereby completing the circuit and allowing current to flow through the LED strip.

The effectiveness of this setup is derived from the MOSFET's ability to act as a switch controlled by the gate voltage. With no voltage on the gate, the channel remains non-conductive; hence, the circuit is open and no current flows through the LED strip. Application of a sufficient voltage (approximately 3.3V from the Pico W GPIO) to the gate allows for the formation of the conductive channel, thus closing the circuit and enabling current flow.

The circuit's functionality was verified through a series of tests where the GPIO22 output was alternated between high and low states, corresponding to turning the LED strip on and off, respectively. These tests confirmed that the circuit performed as expected, with the MOSFET effectively controlling the connection between the LED strip and ground based on the GPIO signal. During testing, it was observed that the immediate activation of the LEDs upon application of the 12V supply was mitigated by ensuring correct MOSFET wiring and operation as described. Measurements of voltage levels at the gate, source, and drain were consistent with theoretical expectations, affirming the accuracy of the circuit design and the reliability of the components used.

The implemented design effectively demonstrates the utility of using an N-channel MOSFET for controlling high-power devices like LED strips with low-voltage digital signals from microcontrollers. This configuration not only provides efficient switching capabilities but also simplifies the interfacing between high and low-voltage systems in embedded applications. Further optimization of the circuit could involve the integration of additional control features such as dimming and flashing, which could be achieved by implementing PWM (Pulse Width Modulation) techniques through the Raspberry Pi Pico W.

1.5 Motor Control Circuit

The motor control circuit is similar to the LED control circuit, with the primary difference being the use of a CrocSee 12VDC diaphragm pump as the load. The CrocSee pump is a high-quality diaphragm pump designed for hydroponic systems, aquariums, and other applications requiring water circulation. The pump operates on a 12V DC power supply and is controlled by a Raspberry Pi Pico W microcontroller through an IRLZ34N N-channel MOSFET. The circuit configuration is based on a low-side switching design, allowing the microcontroller to modulate the pump's operational state by controlling the negative power line.

A 1N4001RLGOSCT diode [12] is used in this circuit as a flyback diode to protect the MOSFET from voltage spikes generated by the pump's inductive load. The flyback diode is reverse-biased during normal operation, allowing current to flow through the pump. When the pump is turned off, the diode becomes forward-biased, providing a path for the voltage spike to dissipate harmlessly.

This circuit is controlled via GPIO21 on the Raspberry Pi Pico W, which is connected to the gate terminal of the IRLZ34N MOSFET. The source terminal of the MOSFET is connected to ground, while the drain terminal is connected to the ground terminal of the CrocSee pump. The positive terminal of the pump is connected to the 12V power supply, ensuring the necessary driving voltage for the pump's operation. By modulating the GPIO21 output between high and low states, the microcontroller can control the pump's operational state, enabling water circulation within the hydroponic system.

Some pumps can change direction by reversing the polarity of the power supply, but the CrocSee pump used in this circuit operates in a single direction. The pump's operational state can be controlled by turning it on and off as needed to maintain optimal water circulation within the hydroponic system. The circuit's functionality was verified through a series of tests, confirming that the pump could be controlled effectively using the Raspberry Pi Pico W microcontroller. The flyback diode successfully protected the MOSFET from voltage spikes generated by the pump's inductive load, ensuring the circuit's reliability and longevity.

1.6 Component Testing

1.6.1 BMOUO 12V AC-to-DC Power Supply

The 12V30A power supply unit (PSU) was tested to verify its output voltage and current ratings. The PSU was connected to a digital multimeter to measure the output voltage and current under various load conditions. The voltage and current readings were compared to the manufacturer's specifications to ensure that the PSU was operating within the specified range. The test results confirmed that the PSU was delivering the required 12V voltage, making it suitable for powering the hydroponics system. The PSU's stable output voltage and current ratings ensure reliable operation of the system's components, providing the necessary power for optimal performance. The system has two voltage requirements that the power supply provides. The Raspberry Pi Pico W and sensors can only handle up to 5V. The water pump and LEDs required a higher voltage of 12V. To provide both voltages from a single outlet, a BMOUO 12V AC-to-DC power supply and

DORHEA C120503 12V to 5V DC converter are used. Further more, a FILSHU 10A 250V Power Socket Inlet Switch with a Detachable 3-Prong Power Cord connects the power supply to the 120V wall outlet. As shown in Figure 3, the smaller buck converter was attached to the power supply.

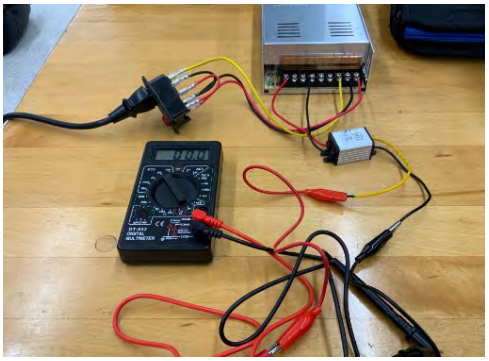


Figure 3: Power Supply Test Circuit

With the components all wired, the power cord was plugged into a power outlet, and the 12V power supply was turned on. A green light on the power supply turned on indicating that the unit has not experienced a short or failure. When we measured the output voltage, we got a reading of 12V. This confirmed the operation of the power supply. The multi-meter was then used on the Buck Converter and a steady measurement of 4.9V was observed. This final measurement showed that the entire power supply circuit was functional and the other components were ready to be powered.

1.6.2 DORHEA C120503 12V to 5V DC Converter

The DORHEA C120503 12V to 5V DC converter was tested to verify its voltage conversion capabilities and efficiency. The converter was connected to a 12V power supply and a digital multimeter to measure the output voltage and current. The voltage and current readings were compared to the manufacturer's specifications to ensure that the converter was operating within the specified range. The test results confirmed that the converter was converting the 12V input voltage to 5V, making it suitable for powering the Raspberry Pi Pico W microcontroller and other 5V sensors. The converter's efficient voltage conversion and stable output ensure reliable operation of the microcontroller and other components, facilitating seamless integration into the hydroponics system. The test was repeated twice: first with the lab DC power supply and then with the PSU to ensure that the converter was functioning correctly under different load conditions.

1.6.3 FILSHU 10A 250V Power Socket Inlet Switch

The FILSHU 10A 250V power socket inlet switch was tested to verify its functionality and safety features. The switch was connected to the 12V30A power supply and a digital multimeter was connected to the power supply. The switch was then toggled on and off to verify that it could control the power supply's output. The test results confirmed that the switch was functioning correctly, allowing for easy power management and control of the hydroponics system. The switch's safety features and robust construction ensure reliable operation and user-friendly control of the system's power supply. It was found that it takes roughly 10 seconds from switching the inlet switch off until the power fully dissipated from the PSU.

1.6.4 Raspberry Pi Pico W

The Raspberry Pi Pico W [14] had to undergo extensive testing to ensure that it was functioning correctly and could handle the demands of the hydroponics system. The Pico was tested by connecting it to a computer via micro-USB and running a series of test scripts to verify its GPIO, ADC, and I2C functionality. The onboard temperature sensors were not functioning correctly, but this was not concerning because external DHT22 sensors were already incorporated into the system. The Pico was also tested with the OLED display to verify that it could communicate with the display and output data correctly. The test results confirmed that the Pico was functioning as expected, providing the necessary processing power and I/O capabilities for the hydroponics system. The Pico's compact size and low power consumption make it an ideal choice for embedded applications, ensuring efficient operation and reliable performance.

The I2C and UART communication protocols were tested using the Pico's GPIO pins. The I2C protocol was tested by connecting the Pico to an OLED display and running a MicroPython script to display text on the screen. The UART protocol was tested by connecting the Pico to a computer via USB and running a script to send and receive data over the serial connection. The tests confirmed that the Pico's communication protocols were functioning correctly, enabling seamless data exchange between the microcontroller and external devices. The Pico's versatile communication capabilities make it well-suited for interfacing with a wide range of sensors and peripherals, enhancing the system's functionality and flexibility.

1.6.5 DHT22 Temperature and Humidity Sensor

The DHT22 sensor was tested to verify its accuracy and reliability in measuring environmental conditions. The sensor was connected to the Raspberry Pi Pico W microcontroller, and a MicroPython script was run to read the temperature and humidity values. The sensor was placed in various environments with known temperature and humidity levels to compare the readings and assess the sensor's performance. Boveda humidity control packets [3] were used to verify the humidity readings when in an enclosed space with controlled humidity. The test results confirmed that the DHT22 sensor was accurate and reliable in measuring temperature and humidity, providing valuable data for monitoring the hydroponics system's growing environment. The sensor's stable performance and consistent readings ensure precise monitoring of environmental conditions, enabling users to optimize growth conditions and ensure healthy plant development.

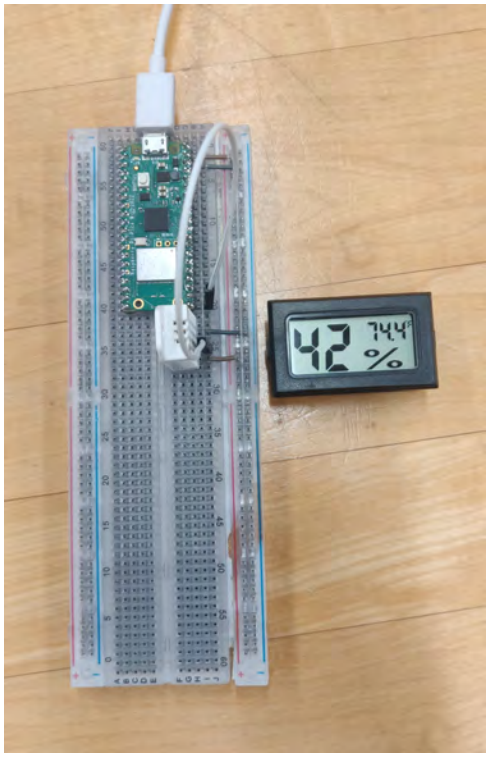


Figure 4: DHT-22 Test Circuit

The Luli Hydroponic System is a host to 5 seamlessly integrated sensors. Each sensor plays a critical role in successful hydroponics, and after acquisition of the components, their functionality and accuracy was tested. The DHT-11 sensor was chosen due to its dual utility as a humidity and temperature sensor. This reduces the number of GIO pins needed from the Pico microcontroller. According to the datasheet provided by Mouser electronics [**?**], the DHT-11 sensor can measure temperatures within an accuracy of $\pm 2°C$. Similarly, humidity measurements have an accuracy within $\pm 5\%$. In order to test functionality, the sensor's VDD pin is connected to the VBUS of the microcontroller. The sensor's data prong is linked to the Pico's GIO 1 pin, in conjunction with a $5k\Omega$ pull-up resistor. As shown in Figure 4, the test circuit is completed by connecting the sensor and the microcontroller's ground together. In order to program the microcontroller, the coding IDE Thonny was used and the software was developed in microPython. The test first defines the data pins connected to the DHT-11 sensors. A while loop is used to continuously poll the sensors until a successful measurement is conducted and the output is printed. A short delay of 3 seconds is provided inline with the sensor's minimum measurement rate.

```
MPY: soft reboot
Starting DHT11.
Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...Measuring.
...
```

Figure 5: DHT-11 Failed Test

The first round of testing on the DHT-11 proved unsuccessful, and as shown in Figure 5, a Timeout error was encountered when attempting to gather measurements. To determine the source of the fault, it was important to test the remaining supply of DHT-11 sensors with the test circuit before any modifications to the code or circuit were made. After swapping each DHT-11 sensor, it was discovered that each behaved in the same manner and the timeout error was not resolved. The timeout error is typically caused when a sensor fails to respond with data within an expected period of time. Therefore, the delay within the test code was reduced to the minimum delay of 1s, which is specified in the DHT-11 datasheet, and then adjusted far longer than needed. Neither change produced a successful result and therefore attention was returned again to the hardware.

In an attempt to further pinpoint the cause of the issue, it was then assumed that the sensors used could in fact be malfunctioning. To test this theory, an alteration to the test circuit was made by replacing the sensor with its more reliable version, the DHT-22. After swapping the sensors in the test circuit, the code was run in Thonny and an attempt to gather environmental measurements was attempted. The exchange in sensors proved to be successful and a temperature reading was returned. Depicted in Figure 6, the DHT-22 sensors returned humidity and temperature data after a series of readings and it was therefore decided to alter the original design of the system to include DHT-22 sensors instead of the DHT-11 sensors.

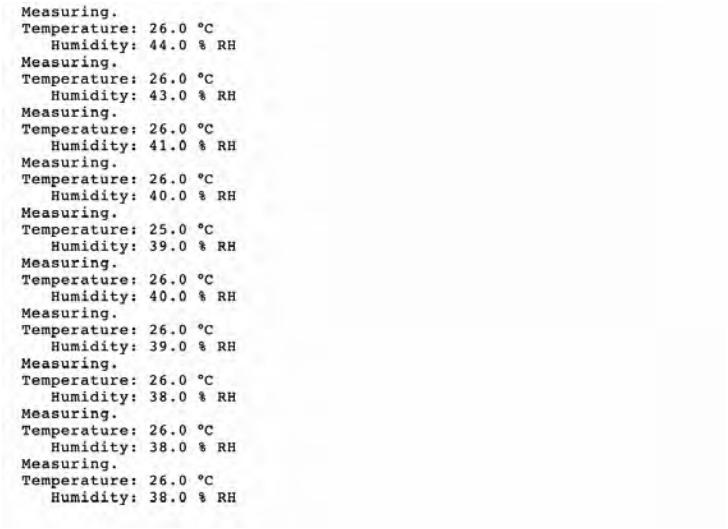


Figure 6: DHT-22 Successful Readings

1.6.6 LTR-390UV-01 ALS + UV Sensor

The LTR-390UV-01 UV [9] sensor was tested to verify its accuracy and reliability in measuring UV light intensity. The sensor was connected to the Raspberry Pi Pico W microcontroller, and a MicroPython script [4] was run to read the raw UV light intensity values. The sensor was placed in ambient room lighting and then under dedicated UV LEDs to compare the readings and assess the sensor's performance. The sensor was not sensitive enough to detect any UV light from a source more than a few inches away, but after increasing the gain to the maximum value it supported (B.7) the sensor was able to detect the UV light from the LEDs. The test results confirmed that the LTR-390UV-01 sensor was accurate and reliable in measuring UV light intensity, providing valuable data for monitoring the hydroponics system's UV lighting system. The sensor's stable performance and consistent readings ensure precise monitoring of UV light levels, enabling users to optimize growth conditions and ensure healthy plant development.

1.6.7 CrocSee 12VDC Diaphragm Pump

The CrocSee 12VDC diaphragm pump [5] was not the original pump selection for this system. The original water pump, a Gikfun 12V DC dosing pump with peristaltic dosing head [6] was used. This original pump, however, sustained water damage 7 and longer functions. The CrocSee pump was selected as a replacement due to its higher flow rate and compatibility with the system's 12V power supply. The pump was tested to verify its flow rate and operational performance in circulating water within the hydroponics system. The pump was first connected directly to the 12V power supply when both inlet/outlet tubes were submerged in a container of water. The pump was then connected to the Raspberry Pi Pico W microcontroller through an IRLZ34N MOSFET to enable digital control of the pump's operational state. The pump was tested with a MicroPython script to turn it on and off, verifying that the pump could be controlled effectively using the microcontroller. The test results confirmed that the CrocSee pump was suitable for circulating water within the hydroponics system, providing the necessary flow rate and operational reliability for optimal plant growth. This test also served as the basis for the IRLZ34N MOSFET test, which was successful in controlling the pump's operational state.

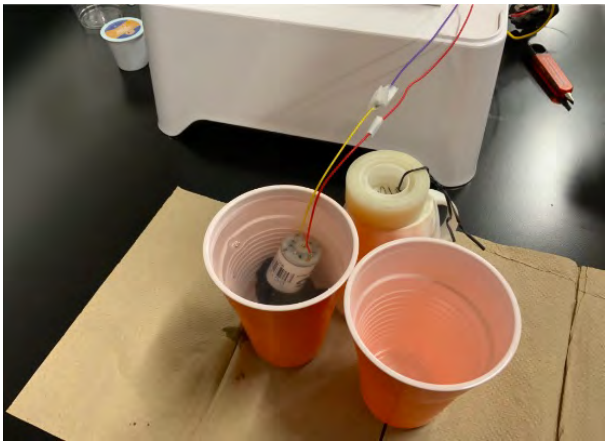


Figure 7: Failed Pump Test

1.6.8 IRLZ34NPBF MOSFET

The IRLZ34NPBF MOSFET was selected due to its high drain-to-source voltage rating at 55V and low on-resistance (B.3). The MOSFET was tested to verify its switching capabilities and operational performance in controlling high-power devices like the CrocSee 12VDC diaphragm pump. The MOSFET was connected to the Raspberry Pi Pico W microcontroller through a 220µF capacitor and a 1N4001RLGOSCT flyback diode to protect the MOSFET from voltage spikes generated by the pump's inductive load. The MOSFET was then tested with a MicroPython script to turn the pump on and off, verifying that the MOSFET could control the pump's operational state effectively. The test results confirmed that the IRLZ34NPBF MOSFET was suitable for controlling high-power devices like the CrocSee pump, providing reliable switching capabilities and robust performance in the hydroponics system. The MOSFET's low on-resistance and high voltage rating ensure efficient operation and safe control of the pump, enabling precise water circulation within the system. Each MOSFET would be used as a low-side switch to control the pump's and LED's respective operational states, allowing for digital control of the loads' power connections.

Low-side switching using an IRLZ34N MOSFET involves connecting the source terminal to ground and the drain terminal to the load, with the other side of the load linked to a power supply's positive terminal. This configuration allows for straightforward control of the MOSFET by applying a voltage to the gate relative to its source. When a sufficient gate-to-source voltage is provided—typically around 2V for the IRLZ34N—the MOSFET conducts, allowing current to flow from the power supply through the load and into the drain, activating the load. Deactivation occurs when the voltage is removed from the gate, stopping current flow and thus powering down the load. This method is favored for its simplicity and cost-effectiveness, requiring fewer components and less complex drive circuitry compared to high-side switching. However, it is crucial to address potential ground reference issues and manage voltage spikes from inductive loads with protective components like flyback diodes to ensure robust operation.

1.6.9 1N4001RLGOSCT-ND Flyback Diode

The 1N4001RLGOSCT-ND [12] diode was intended to be placed in reverse-bias across the CrocSee pump to protect the IRLZ34N MOSFET from voltage spikes generated by the pump's inductive load. The first attempt at testing the diode revealed that it was accidentally placed in forward-bias, causing the diode to burn out. A new, identical diode was then correctly placed in reverse-bias across the pump, and the test was repeated. The test results confirmed that the diode was functioning correctly, providing a path for the voltage spike generated by the pump's inductive load to dissipate harmlessly. The diode's reverse-bias configuration ensures that it remains non-conductive during normal operation, allowing current to flow through the pump. When the pump is turned off, the diode becomes forward-biased, providing a path for the voltage spike to dissipate safely, protecting the MOSFET, and ensuring the circuit's longevity. The diode's protective function safeguards the MOSFET from potential damage due to voltage spikes, ensuring reliable operation and robust performance of the hydroponics system.

1.6.10 Ultra-Sonic Sensor HC-SR04

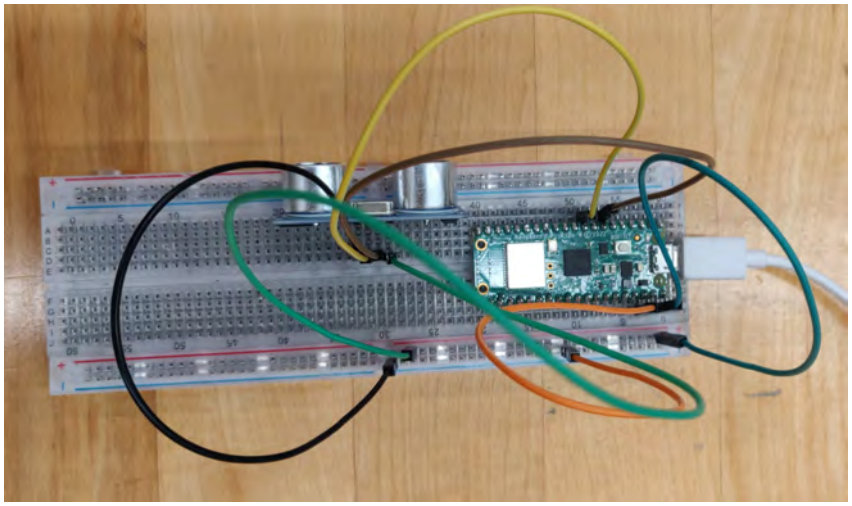


Figure 8: Luli Hydroponic System Housing

The HC-SR04 ultrasonic sensor was placed on top of the reservoir tank and takes measurements of the solution's level. The sensor is accurate within 3mm and has a practical range between 2cm and 80cm [**?**]. Two tests were conducted on this sensor. First the unit was tested for functionality. In order to accomplish this, the ultrasonic sensor was interfaced to the Pico by first connecting the VCC pin to the VBUS output of the Raspberry Pi Pico. This provides the unit with the 5V necessary to drive it. The trigger pin was connected to GIO pin 5 and the echo pin was interfaced with GIO pin 6. The microPython program activates the trig pin for 10 microseconds and the echo pin receives the returning ultrasonic pulse, which is transmitted to the Pico for analysis and conversion. Once the data from the ultrasonic pulse is received by the microcontroller, the data must be converted.

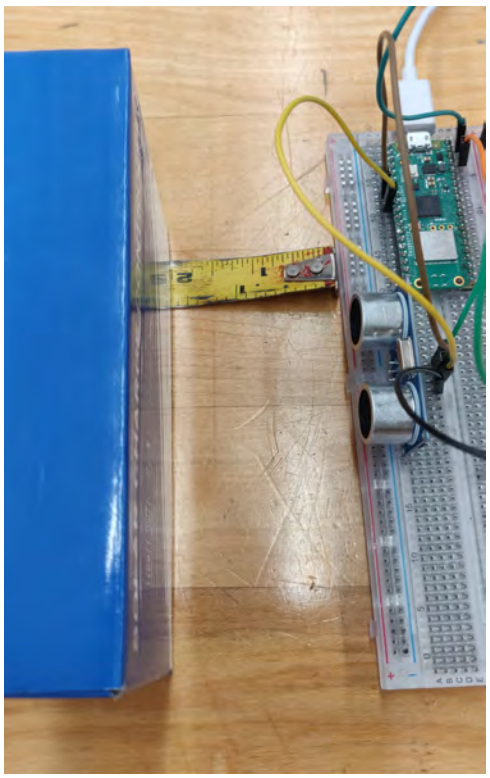


Figure 9: HRSR04 Accuracy Testing

In order to confirm that the sensor is reading data correctly, a function was written that first sends a logical 1 value to the trigger pin, and if successful, the transmitter will emit an ultrasonic pulse that the receiver will detect. The echo pin will go high at the end of the transmission and remain high until an ultrasonic pulse is detected by the receiver. Once the function detects the low signal from the echo pin, the difference between the signalOn and signalOff is calculated to determine the period, which is measured in microseconds, that the sonic signal took to travel between two points. This time period is then converted by multiplying by the distance sound travels in a micro second:

$$\text{Distance} = \frac{\text{Time Duration} \times 0.0343}{2} \tag{1}$$

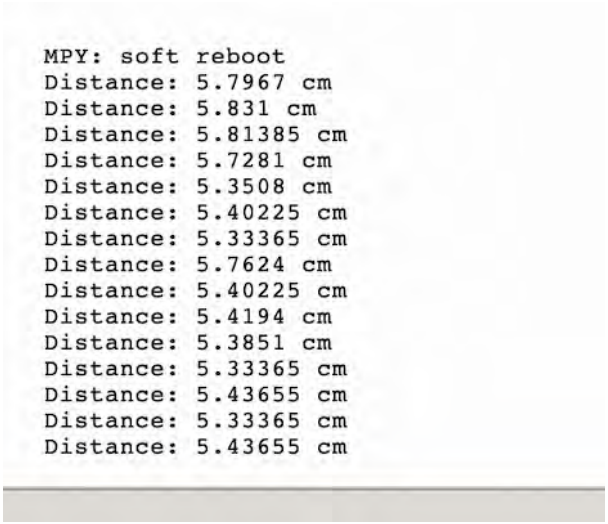


Figure 10: HCSR04 Measurement Results

When the test code was run, the unit appeared to be operational and a series of measurements were output to the terminal. This confirms that the unit is at least functional, and sends a signal when triggered. In order to test that the data printed was accurate, an object was placed 5.0cm apart from the sensor. A measurement within 3mm would be considered accurate and within the manufacturer specifications [**?**]. After running the test code a second time, it was determined that the measurements were sufficiently accurate with a small spike in inaccuracy in the initial rounds of the measurement loop. Overall all the measurements, shown in Figure 10, were within 3-5mm of the sensor. This is more than acceptable when we consider that the distance of 5.0cm was measured from the front of the receiver's housing and the actual hardware that detects the echo pulse recedes an unknown distance within the housing.

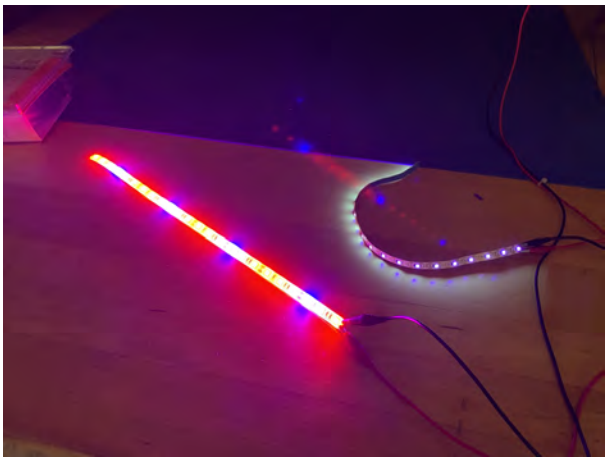
1.6.11 74LS151N 8:1 MUX

The multiplexer [15] was tested by connecting it to the ADC0 pin on the Raspberry Pi Pico W and using the GND, 3.3V, and 5V outputs from the Pico as known input voltages. The multiplexer was then used to switch between these inputs and output the selected voltage to the ADC0 pin. The output was then read using a MicroPython script (A.1.10) to verify that the multiplexer was functioning correctly. The test results confirmed that the 74LS151N 8:1 MUX was operating as expected, allowing for the selection of different input voltages and outputting the desired voltage to the ADC0 pin.

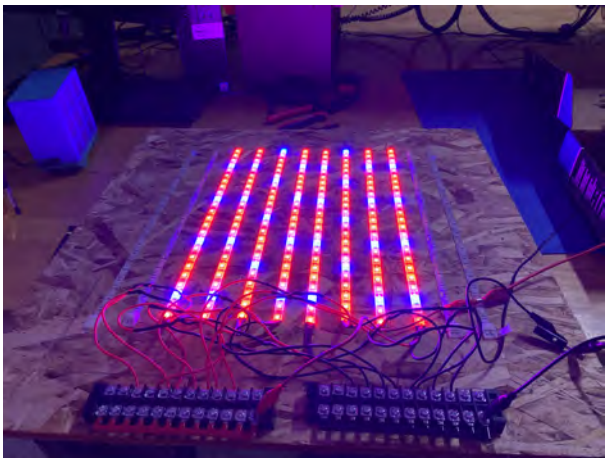
Unfortunately, during the final few weeks before the culmination of the project, the D0 and VCC pins on the MUX broke off, rendering the multiplexer unusable. This issue was identified during the final integration phase, and the decision was made to proceed without the multiplexer. This freed up GPIO pins that were originally used for select, enable, and the data line which were now replaced with the data lines of the sensors directly.

1.6.12 60W Full Spectrum Grow Lights

The 60W full spectrum grow light LED strips [7] are rated for 12V5A producing 60 Watts of power. This is a significant upgrade from the originally purchased LED strips [8] that were only rated for 5V2A. The purchasing of the original LEDs was an oversight because the power supply was rated for 12V, and the LEDs were rated for 5V. The new LEDs were tested by connecting them directly to a DC power supply and observing the light output and current usage. The test results confirmed that the LEDs were functioning correctly, providing the necessary full spectrum light for plant growth. The LEDs were then connected to the Raspberry Pi Pico W microcontroller through an IRLZ34N MOSFET to enable digital control of the LED strip's operational state. The LEDs were tested with a MicroPython script to turn them on and off, verifying that the MOSFET could control the LED strip's operational state effectively. The test results confirmed that the 60W full spectrum grow lights were suitable for providing the necessary light intensity for plant growth, enabling optimal photosynthesis and healthy plant development. The LEDs' high power output and full spectrum light coverage ensure robust growth conditions and promote healthy plant development within the hydroponics system. The difference in output is noticeable when comparing the two LED strips (11a).



(a) 60W Full Spectrum LEDs (left) vs. 10W UV LEDs (right)



(b) Successful Parallel Wiring of LED Array

Figure 11: Comparative LED illumination and array configuration

1.6.13 pH Sensor - PH4502C

pH	Voltage [V]
4	3.071
7	2.535
10	2.066

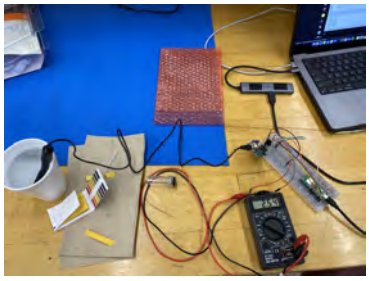
Table 1: PH4502C pH to Voltage Conversion

Figure 12: PH4502C pH Sensor [1]

The PH4502C [13] is designed to output the following voltages shown in Table 1. This data was provided by the manufacturer and was used to calibrate the pH sensor. An extensive calibration process was conducted for the PH4502C sensor module to ensure accurate pH readings. The sensor, equipped with two potentiometers for offset and slope adjustments, was designed to output a voltage correlating to the detected pH level, with a specified range of 0-5V.

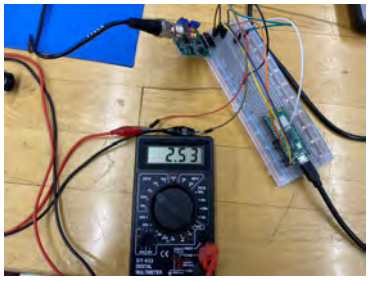
Initially, difficulties were encountered in the calibration process due to output voltage readings exceeding the safe input range of the Raspberry Pi Pico's analog-to-digital converter (ADC). A voltage divider was implemented to attenuate the sensor's voltage output. Subsequent readings from the sensor module were consistently registering maximum ADC values, indicative of an over-voltage condition. It was concluded that the direct output from the sensor was unsuitable for direct interfacing with the Pico's ADC. Following the integration of a voltage divider into the circuit, the output was reduced to acceptable levels within the Pico's ADC range of 0-3.3V (13c). However, the initial calibration attempt, while using the voltage divider, produced a reading that did not align with the standard pH buffer solution. Consequently, the calibration process was refined by removing the voltage divider and reading the output directly from the sensor module (13a), which resulted in a more stable and accurate voltage reading.

The offset potentiometer was then carefully adjusted while the sensor was immersed in a pH 7 buffer solution. The voltage was monitored using a multimeter, and the potentiometer was tuned until the output voltage closely matched the expected 2.51V as indicated on the product page for a pH 7 solution. This calibration point was further verified through a MicroPython script, which confirmed a pH reading of 7, correlating to the observed voltage of 2.5V. Finally, the pH was tested using litmus strips (13b) to verify the sensor's accuracy in detecting the pH level of the solution. The sensor was deemed accurate and reliable in measuring pH levels, providing valuable data for monitoring the hydroponics system's nutrient solution. The sensor's stable performance and consistent readings ensure precise monitoring of the nutrient solution's pH, enabling users to optimize nutrient levels and ensure healthy plant growth.



(a) pH Sensor Test Setup

(b) Litmus Strip Verification



(c) 2.5V Voltage Verification

Figure 13: PH4502C pH Sensor Testing

An observation was made during the calibration that a red LED indicator on the sensor module was activated when the voltage dropped below 4V. It was inferred that this LED served as an indicator for out-of-range voltage conditions. The sensor was adjusted to maintain the output voltage above this threshold to prevent the activation of the LED indicator. Once the sensor output was adjusted to the correct voltage for a pH 7 solution, the calibration was deemed successful. The calibration state was documented as stable, and the expectation was set that, barring any significant handling or environmental changes, recalibration would not be necessary before each use.

1.6.14 OLED Screen

The screen chosen for our design is the 2.42in OLED LCD display produced by DWEII Electronics. In order to test this screen, we first wired the component to the Raspberry Pi Pico W. The VCC pin on the OLED display was connected to the VBUS of the microcontroller and draws a voltage of 3.3v. When choosing pins on the Raspberry Pi Pico, it was important to note which protocols were supported, and for this test, pins 10-15 were chosen since they support SPI communication. GIO pin 10 was defined as the system clock output and set to a baud rate of 1MHz. The screen communicates with the Pico using SPI communication and the screen acts as the slave unit. To interface the communication between devices pin 11 on the Pico was used as the MOSI port. The reset pin of the screen was connected to GIO pin 14, and pin 13 was connected to the CS pin of the screen. The test circuit was completed by connecting the ground of the screen to the Pico's ground pin. To drive the circuit, the code, shown in appendix , was written to display a simple "Successful" message to the screen.

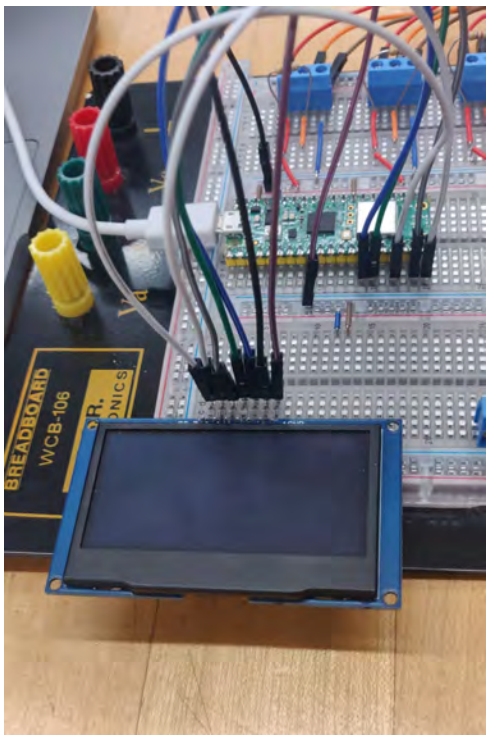


Figure 14: OLED Display Test Circuit

The first attempt to drive the screen was unsuccessful. When the code was run, the screen flickered for a brief moment before it turned off and remained black. Since we observed a change in the screen, it was likely that a wire was incorrectly placed or a component was not fully driven into the breadboard. First we ensured that the connections between the components were secured and correct. After checking connections, the board yet again failed to remain on. Next we checked the headers that were soldered onto the OLED screen. Under closer examination, it was discovered that a pin was not fully soldered. The joint had formed a cold joint, and when moved, we observed that there was space between the cold joint and the VCC pin of the screen. Careful attention was given to each header pin to ensure that no other pins were loose, and the headers were re-soldered on with better connections. Once that error was fixed, the program was run, and on the third attempt, the screen turned on and the success message was displayed. Figure 15 shows the Pico printing the message to the screen and confirms that the unit is functional.

Figure 15: Successful OlED Screen Test

1.7 Printed Circuit Board Design

The hydroponics system is powered by a 12V30A power supply with a maximum current draw of 5A. Temporary prototyping solutions such as a solderless breadboard were used to test the system's functionality. However, to ensure long-term reliability and ease of maintenance, a custom-designed printed circuit board (PCB) was developed to house the system's components. The PCB design process involved creating a schematic diagram, laying out the components, routing the traces, and generating the Gerber files for manufacturing. The PCB design was meticulously crafted to accommodate the system's sensors, microcontroller, motor control circuit, and power supply connections. The resulting PCB design provides a compact and organized layout that simplifies the assembly process and enhances the system's overall reliability.

The schematic and PCB were designed using free open-source software, KiCad. KiCad is a powerful EDA (Electronic Design Automation) tool that offers a comprehensive suite of features for designing schematics, laying out PCBs, and generating manufacturing files. The schematic design process involved creating symbols for each component, connecting the components with wires, and adding labels for easy identification. The schematic served as the foundation for the PCB layout, guiding the placement of components and the routing of traces. The PCB layout was carefully planned to ensure optimal component placement, signal integrity, and thermal management. By organizing the components in a logical and compact manner, the PCB design maximizes space utilization and minimizes signal interference, resulting in a clean and efficient layout.

The PCB was milled in-house using an LPKF ProtoMat Circuit Board Plotter. The milling process involved loading a blank copper-clad board into the plotter, importing the Gerber files generated from KiCad, and executing the milling operation. The LPKF ProtoMat precisely etched the traces and cut out the board outline, resulting in a custom PCB ready for component assembly. The milled PCB was inspected for any defects or errors, and the components were soldered onto the board following the assembly guide. The completed PCB was then tested to verify its functionality and ensure that all components were properly connected. The material for the PCB was 1/2 oz FR4 copper-clad board, which provides excellent thermal conductivity and electrical insulation properties.

The current carrying capacity of a printed circuit board (PCB) trace can be estimated using the formula from IPC 2221 [2], which is given by:

$$I = k \cdot A^{0.44} \cdot (\Delta T)^{0.725} \tag{2}$$

where:

- I is the maximum current in Amperes (A) that the trace can safely carry.
- A is the cross-sectional area of the trace in square millimeters (mm^2), which can be calculated as the product of the trace width and thickness.
- ΔT is the temperature rise above ambient in degrees Celsius (°C) that is considered acceptable for the application.
- k is a constant that depends on the location of the trace on the PCB. It is 0.024 for internal traces or 0.048 for external traces.

According to the provided specifications, for a trace thickness of 18µm on an external layer, the required trace width calculated by the IPC 2221 formula to carry a current of 5A with a temperature rise of 10°C is approximately 211 mils. With this calculated, special trace widths were defined in KiCad to automatically make the 12V5A lines 215mil wide, the 5V lines 50mil wide, and the new default setting 20mil wide. This ensures that the PCB can safely carry the required current without overheating or causing damage to the components.

Multiple through-hole screw terminal blocks were used for the sensors, motor, and power connections to facilitate easy assembly and maintenance. The screw terminals provide a secure and reliable connection for the wires, allowing for quick installation and removal of components. This also allows for distanced connections to the PCB which enables the

LEDs, DHT sensors, water pump, and power supply to be placed in different locations. These terminal blocks were strategically placed on the PCB to ensure optimal wiring organization and accessibility. The PCB design was meticulously crafted to accommodate the system's components and facilitate easy assembly and maintenance. The resulting PCB layout (16) provides a compact and organized configuration that enhances the system's reliability and ease of use.

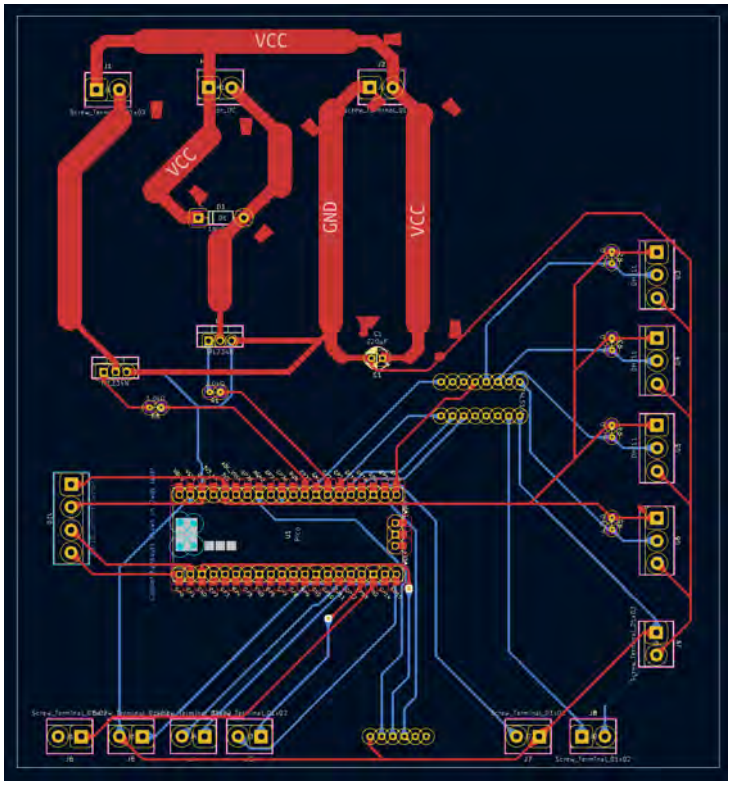


Figure 16: First Iteration PCB in KiCad

After successfully milling the PCB in-house, the components were soldered onto the board following the assembly guide. At this stage, a fatal design flaw was discovered in the PCB layout that prevented several components from being soldered correctly. Several terminal blocks had pads on both layers of the PCB, meaning that it would be nearly impossible to solder the components with such minimal clearance. Additionally, the LPKF ProtoMat Circuit Board Plotter could not make plated through holes, which would have allowed for the components to be soldered on both sides of the board. As such, vias were essentially useless in this design without manually filling them with solder.

These design flaws were identified during the assembly process and were rectified by redesigning the PCB layout (17) to ensure that all components could be soldered correctly. Unfortunately, this setback meant that a compromise had to be made such that only the high-current components were soldered onto the now single-layer PCB. The remaining components were placed on a breadboard and connected to the PCB using jumper wires to ensure that the system could still be tested and demonstrated. The revised PCB layout was milled, and the high-current components were successfully soldered onto the board (18). The system was then tested to verify its functionality and ensure that all components were properly connected. The revised PCB design successfully accommodated the system's components and facilitated easy assembly and maintenance, despite the initial design flaw. The second iteration of the PCB layout was meticulously crafted to ensure optimal wiring organization and accessibility, enhancing the system's reliability and ease of use.

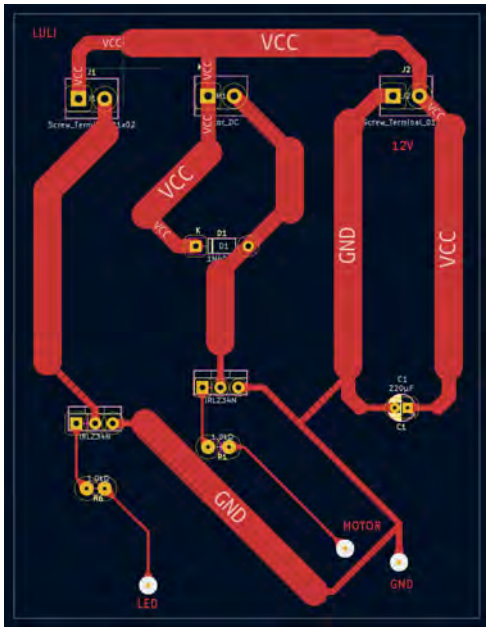


Figure 17: Second Iteration PCB in KiCad

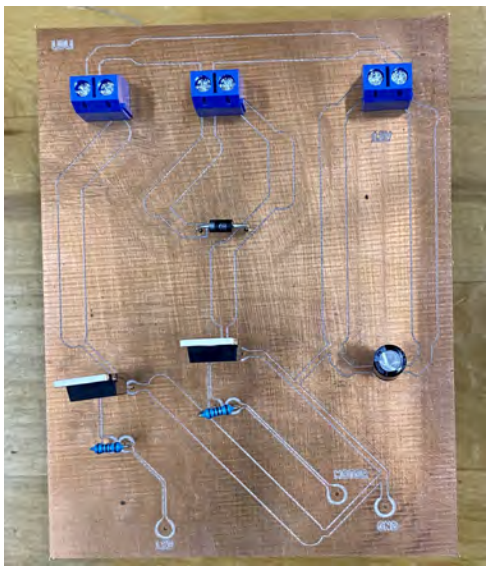
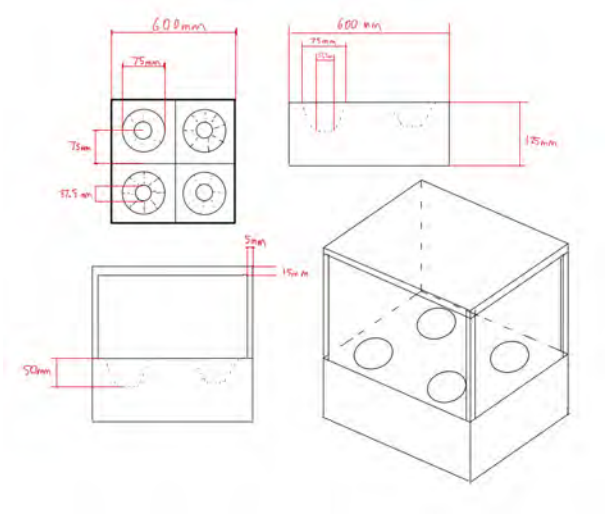


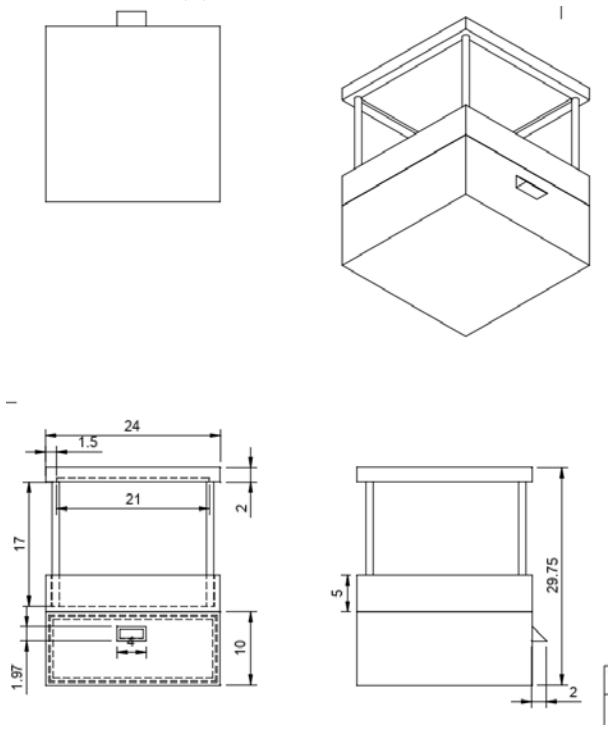
Figure 18: Assembled High Current PCB

1.8 Physical Enclosure Construction

The designed of the physical enclosure was inspired by the need for sufficient area to grow up to 6 plants, a discrete way to route wiring, and a way to protect the electronics from water damage. The first draft of the enclosure was freehand drawn (19a) and then modeled in Fusion 360 (19).



(a) Freehand Design



(b) Fusion360 Dimensioned Drawing

Figure 19: Physical Enclosure Design Blueprints

The dimensioned design in Fusion 360 was rendered to provide a visual representation of the enclosure's layout and size (25). The enclosure was designed to house the hydroponics system's components, including the sensors, microcontroller, motor control circuit, and power supply. The enclosure features a top cover that can be removed for easy access to the system's components, allowing for maintenance and troubleshooting. The enclosure's dimensions were carefully chosen to accommodate the system's components while providing sufficient space for plant growth and water circulation. The enclosure's design ensures that the system's components are protected from water damage and environmental factors, ensuring reliable operation and longevity.

(a) Render: Front View (b) Render: Rear View (c) Render: Underside View

Figure 20: 3D Renderings of the Physical Enclosure

Upon the finalization of the design and renderings, the physical construction took place. The physical structure (**??**) was created from a 4"x8" unit of Oriented Strand Board (OSB). The OSB was cut to size using prior to purchasing and was then transported and finally assembled using and screws and furring strips. The design consisted of two boxes that sat on top of each other: the lower portion houses the reservoir and circuitry and the upper portion holds the plants and plant containers. The enclosure was then painted with a water-resistant sealant to protect the wood from moisture and environmental factors. The top cover was designed to be removable to allow for easy access to the system's components. The 1" diameter PVC used to support the roof was cut to size and attached to the enclosure using screws. The PVC was left uncovered to allow for easy access to the system's components and to allow for access holes to be drilled in the side to route the LED wires through. By hiding the DHT22 and LED strip wiring in the PVC pipes, the system's wiring was kept organized and out of sight, enhancing the enclosure's aesthetic appeal and functionality.

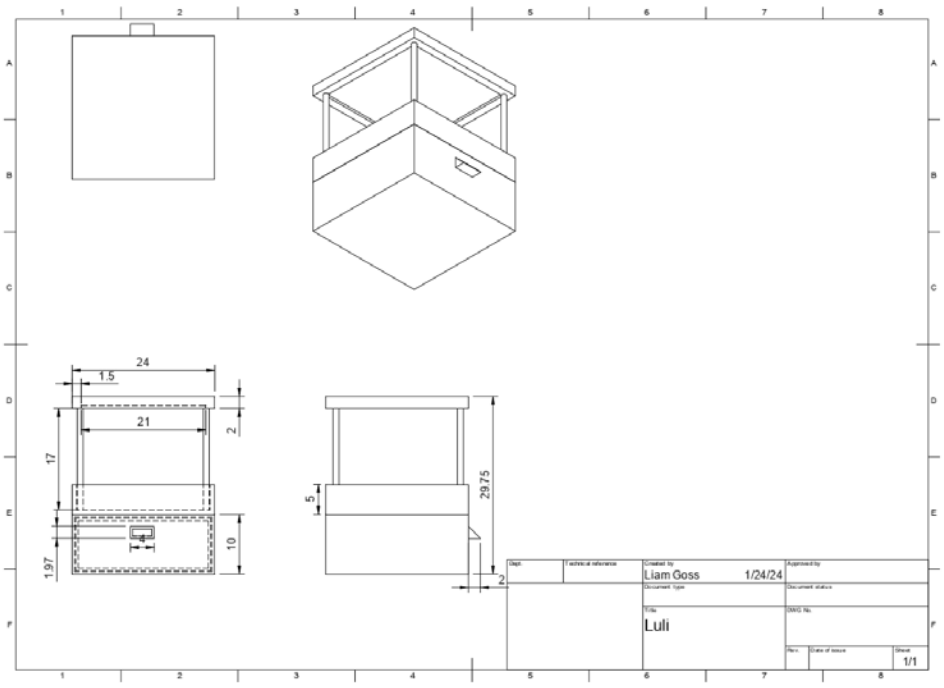


Figure 21: Luli Hydroponic System Enclosure Schematic

The housing of the Luli Hydroponic System is designed to be compact with LEDs and environmental sensors housed in the roof of the enclosure. The body of the system is constructed using an engineered wood material called oriented strand board. This material is often used in large scale constructions for its load bearing properties, due to the lack of internal voids and various orientations of the compressed wood strips. The material is nearly water tight, and the outermost layer requires the application of a water sealant. After cutting the boards to size, three layers of polycrylic sealer were applied to prevent moisture from warping the wood. The housing is split into three individual parts. The design of the bottom enclosure is split into two sections.

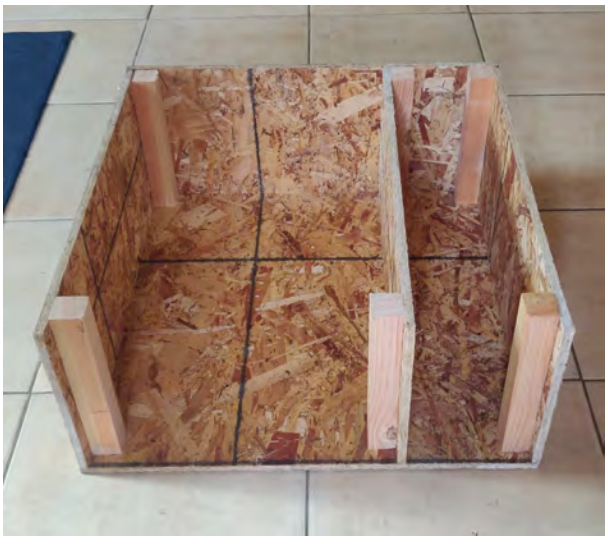


Figure 22: Buttom Section with Seperate Compartments

The largest of the two sections will hold the reservoir, diaphragm pump, HC-SR04 ultrasonic sensor, and pH pump. The reservoir chosen for this design is the 2 gallon Chaplin International Replacement tank. This tank can hold a substantial amount of fertilizer solutions in a low profile body that is 21.6 x 6 x 8.5in. The reservoir is typically used to mix industrial strength agricultural fertilizers, and this re-purposing would fit within the specifications of its original design. This guarantees that the tank can hold fertilizer materials without degrading due to the pH of the growing solutions. Next to the tank will sit the diaphragm pump with its wires running to the driver board located in the adjoining compartment.

Figure 23: Middle Growing Compartement Before Drilling and Adding Growing Trays

The second compartment of the bottom enclosure is considerably more compact with a width of 6 inches. This is where the core of electronics will be housed. Most notably the power supply and Raspberry Pi Pico W. Special care was taken to ensure that the components would fit. For example, it was discovered that the power supply was too long to fit alongside the breadboard and printed circuit board. In order to save on horizontal space, it was concluded that the unit would need to be installed vertically. With this in consideration, the electronic hardware all fit within this section with the wire control from the various components serving as the final cause of concern.

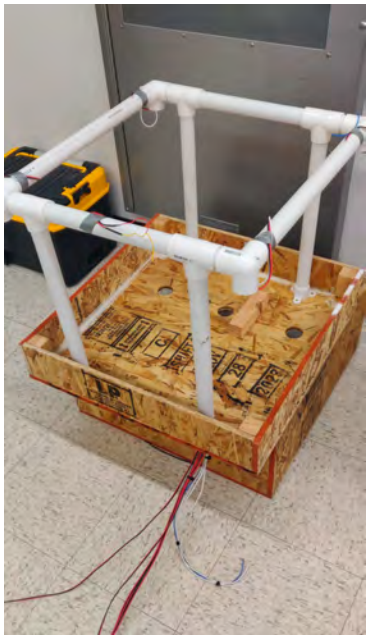
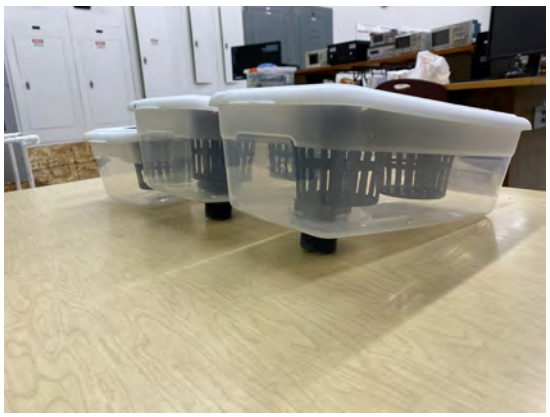


Figure 24: Wiring of The Enclosure with Bulkhead Ports Complete

The base was constructed with four 24x10in boards and a 24x24in floor. The OSB does not handle drilling very well and tends to fray when drilled into. Therefore, before any screw was driven into the board, an appropriate pilot hole was drilled. The edges of the boards are too small to drill into. As shown in Figure 22, a series of 2x2in furring strips were used to give the walls a surface to drill the separate panels together. This technique of drilling pilot holes proved to be very successful and no cracks were developed when connecting the sides together. To further ensure that a failure of the plumping or pump would not flood the electronic components, a ring of caulking was applied to the corners of all surfaces. The construction of the base was simple and no complications were encountered.

(a) Growing Tubs with 4in Grow Baskets



(b) Growing Tubs with Bulkhead Fittings

Figure 25: Growing Tubs

Just above the base, is the growing bed that contains the plant trays, base of the PVC roof supports, and ports for plumbing and electric wires. The growing bed was constructed using 5x24in OSB panels and 24x24in OSB floor. Four furring strips were used to give the panels a foundation to which the panels and floor can be attached to make a square growing tray. Once constructed, a series of holes were drilled into the middle compartment. The first of which were the 3/4in holes for routing piping from the bulkheads of the growing trays back to the solution reservoir. The three growing trays themselves are 6Qt 18x5x8in plastic tubs with a 3/4in hole drilled into the bottom. A PVC bulkhead was attached here to allow for draining of the hydroponics solutions. A pair of 4in holes, depicted in Figure 25a, were drilled to the top to hold the growing baskets. The baskets along with their piping system were installed with no issue.

Figure 26: Growing Compartment Completed with Wiring, Plumbing, and Roof Support

There are seven external components that require wiring back to the Pico and power supply. The LED array and DHT-22 sensors are placed the furthest and would need 18 gauge wires measuring at least 2-3ft. To freely allow these wires to run from the Raspberry Pi Pico W to the sensors, a 1/2in hole was drilled into the floor of the middle housing section and into the PVC leg cap. The wiring for the roof electronics was carefully feed through the middle compartment port and up the leg cap of the PVC pipe. A steel wire was used to grip the end of the electrical wires and pull it through sections of PVC. The DHT-22 sensors were placed at the corners of the roof and the three wires required for the sensor were pulled to each of the four corners. The power and ground wires were colored coded to a traditional black/red, whilst a blue wire was used to indicate a data line. The power wires for the sensors were attached together with zip ties and the data lines were managed in a similar fashion. A small 3/4in hole was drilled into a horizontal PVC pipe of the roof support to supply the LED array with power.

Figure 27: Completed Enclosure

Placing all the electronic components within the enclosure was difficult work that required constant testing after each component was added to the Enclosure. The power supply was first added to the housing, with the 12V DC power supply screwed in vertically to save on space in the circuit section of the bottom enclosure. Furthermore, the PCB and bread board containing the Raspberry Pi Pico W and terminal connections were placed at the forefront of the circuit housing. The wires routed through the PVC pipes were connected to the appropriate terminals on the breadboard. After each collection of sensors were connected, their operations were tested to ensure that they were wired correctly before adding more components. A triangular enclosure was specifically built for the OLED LCD Screen. As shown in Figure 27 the LCD screen was attached to the front of the Luli System with the use of this enclosure. Before placing the reservoir into the larger compartment of the base, a series of holes were drilled into the top of the reservoir. The pH probe was placed in the deepest portion of the reservoir to ensure constant contact with the solution. The other holes were drilled to place the HCSR04 ultrasonic sensor on top of the tank. With these electronics in place, the reservoir was placed into the housing. The wires from these sensors where connected to the bread board and PCB concluding the housing construction and implementation.

1.9 Frontend Design and Development

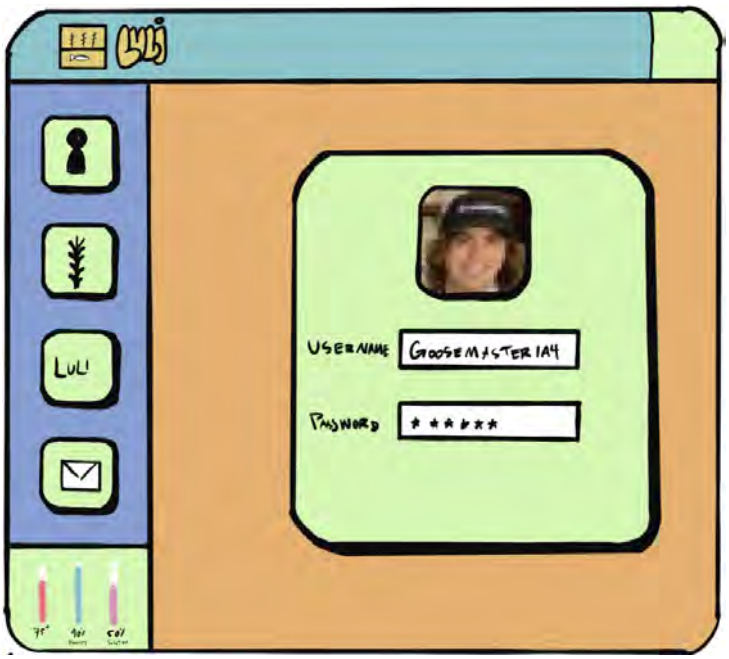


Figure 28: Initial Sketch of The Web Portal

When the Luli Hydroponic system is powered and connected to the internet, the user will interact with the device through the use of the Luli Web Portal. The notable hurdles to horticulturalists new to hydroponics is the high learning curve and constant monitoring of the plants. With this in mind, the design philosophy of the web application places emphasis on user friendly design with a minimalist layout. As shown in Figure 28, the original sketch of the website gives the user a minimal navigation panel. The user would need to first sign in order to see the real time measurements in the bottom right corner of the sketch. The colors scheme was chosen to give the portal a retro appearance mixed with earthy colors.

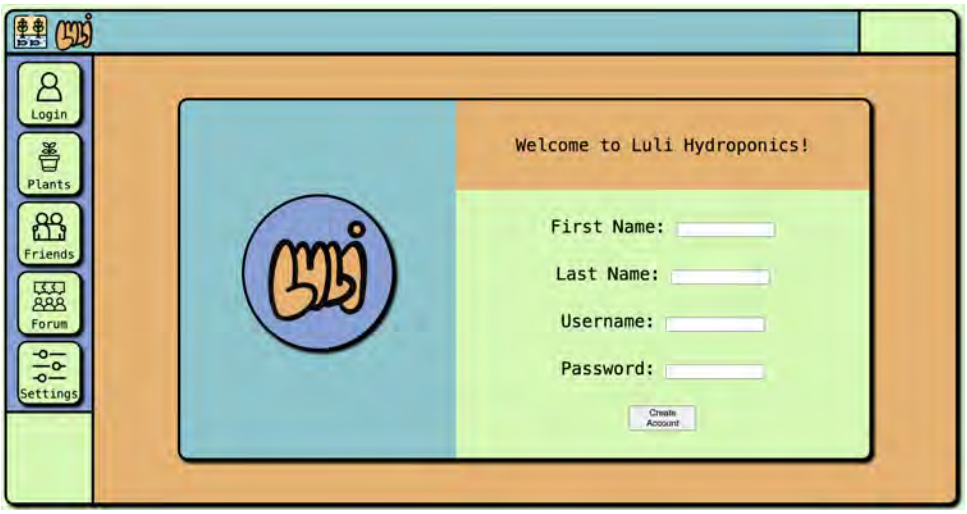


Figure 29: Create Account Portal

The front-end was developed using HTML, CSS, and JavaScript, which adds dynamic elements to the static pages. The implementation of the portal is depicted in Figure 29, and some key changes were made to the original design. Users now have the option to view their friend's list, post to the cite forum, and adjust their system's settings. Most notably, the measurements were moved from the bottom right corner onto their own separate page. As shown in 30, the plant page is neatly organized with the data presented clear and in a manner that is intuitive. The left panel presents the user with real time measurements served by the Flask supported back-end server. The container left of the measurements is a plant info panel that gives the user information about which plants are currently in the enclosure, important dates, and a field indicating which container holds the plant.

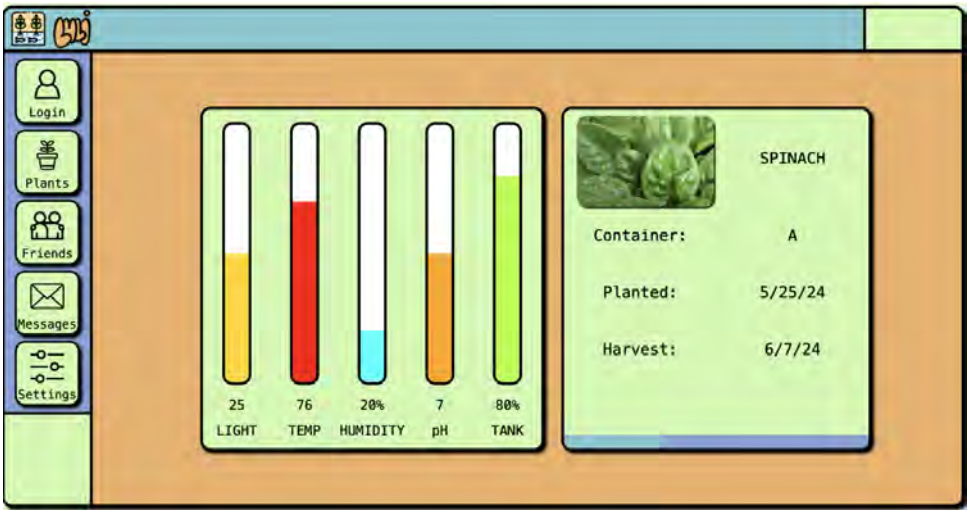


Figure 30: Plant Information Page

The settings page gives the user the ability to modify the default settings of their hydroponic system. From settings, the user can choose how often their plants get watered, the amount of water the plants receive, and the duration of the light cycles. Furthermore, users can adjust the plant info panel from this page to better reflect the plants present in their grow trays.

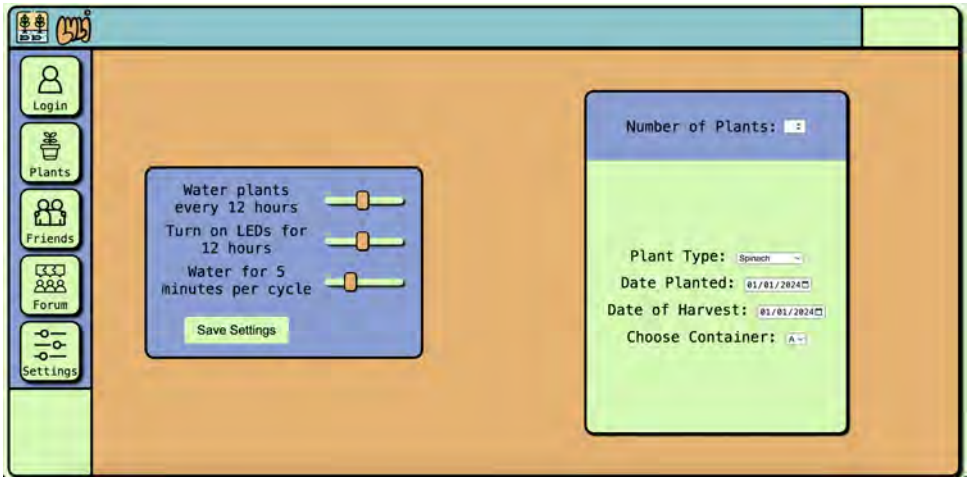


Figure 31: Settings Control Panel

The social component of the Luli Hydroponic System is demonstrated in Figure 32, where users can give updates about their plant through their about section. The Down arrow is a button where users can download each other settings, which is then applied to their system. Each time the settings page is updated, these friend settings are overwritten. The Luli website establishes a platform where users are able to freely connect, share growing techniques, and building communities with open access to nutritious choices.

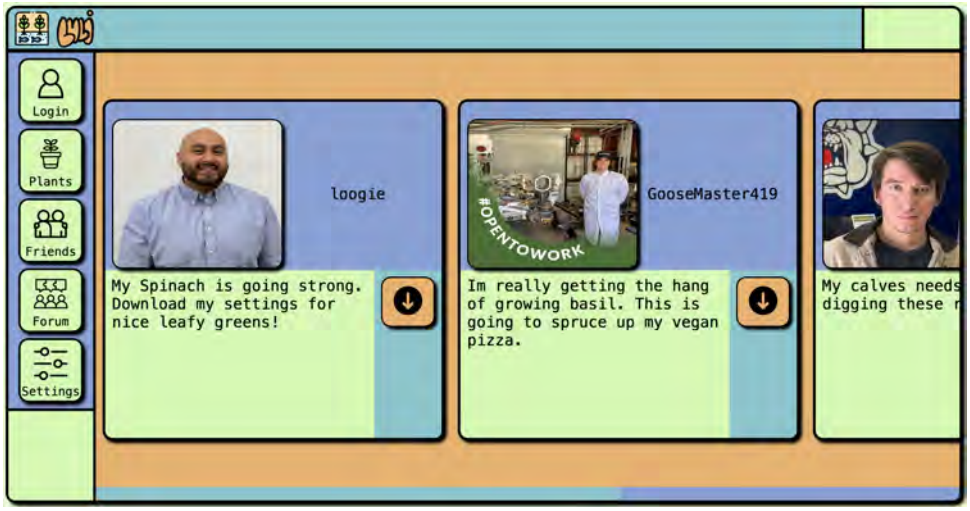


Figure 32: Luli Website Friends Page

1.10 Server Configuration

The central server is configured to house the web application and manage the data from the hydroponics systems. The server runs on a Debian operating system. Debian is a popular and widely used Linux distribution known for its reliability and extensive package management system. Debian offers a wide range of software packages through its repositories. It is favored for server environments due to its reliability and security features, making it a suitable choice for hosting various web applications, services, and data management systems. The entire backend system sits in a 42U server rack (33). The server rack is equipped with a UPS (Uninterruptible Power Supply) to ensure continuous operation in the event of power outages or fluctuations. The server is connected to the internet via a high-speed gigabit ethernet connection, providing reliable connectivity for remote access and data transfer.

(a) Empty Server Rack (Rear)

(b) Empty Server Rack (Side)

(c) Active Server Rack (Front)

Figure 33: 3D Renderings of the Physical Enclosure

Running on the Debian host machine is a Flask server. Flask is a simple and flexible framework for building web applications using Python. It helps developers create websites by providing tools for handling web page requests, organizing URLs, and displaying content. In Flask, routes serve as mappings between specific URLs and Python functions, enabling the application to respond to different requests. You can define custom functions for each URL endpoint, allowing dynamic content generation based on user interactions. Using the `@app.route()` decorator, you specify the URL pattern associated with a particular function. Dynamic URLs can include variable parts, passed as arguments to the associated function, facilitating personalized responses. Within these functions, you can implement logic to generate dynamic content, such as fetching data from databases or processing user inputs. This approach enables the creation of dynamic web applications where each URL endpoint serves specific functionalities, thus offering a tailored user experience.

To enhance network reliability and accessibility for remote services, the deployment and configuration of the No-IP Dynamic Update Client (DUC) were undertaken on the Debian-based system. No-IP is a dynamic DNS service that allows internet users to provide a fixed domain name to their dynamically changing IP address, thereby ensuring that the connection to the server remains uninterrupted despite changes in the network configuration. This is particularly useful for hosting servers, remote access, and other network services that require constant accessibility over the Internet. To integrate No-IP's DUC into the system's services for automatic startup and recovery, a systemd service unit file was meticulously crafted and deployed. This configuration specifies the execution of the No-IP DUC software as a background service, thereby enabling it to update DNS records automatically whenever the system's IP address changes. Through careful examination and troubleshooting of systemd service file settings, potential issues were identified and resolved, ensuring seamless operation. The implementation enhances the system's network reliability and facilitates uninterrupted remote access.

The remote server has two ports exposed to the internet, one for the web application and one for SSH. To reduce the risk of unauthorized access, the SSH port was changed from the default port 22 to a non-standard port. This simple security measure significantly reduces the number of unauthorized login attempts and enhances the system's overall security. The web application port was also changed to a nonstandard port to further reduce the risk of unauthorized access. In addition to altering the default ports for SSH and the web application, two additional security measures were implemented on the remote server: fail2ban and ufw.

Fail2ban is an intrusion prevention software framework that operates by monitoring log files for patterns indicating unsuccessful login attempts or other malicious activity. Upon detection of such patterns, fail2ban dynamically updates firewall rules to block the IP addresses associated with the detected activity. By effectively blocking malicious actors attempting unauthorized access, fail2ban enhances the server's security posture and mitigates the risk of successful brute-force attacks.

UFW, or Uncomplicated Firewall, is a front-end for managing firewall rules in Linux-based systems. It provides a user-friendly interface for configuring firewall settings and managing network traffic. UFW simplifies the process of creating and maintaining firewall rules, enabling administrators to define access policies based on specific criteria such as IP addresses, ports, and protocols. By leveraging UFW to enforce firewall rules, administrators can restrict access to services running on the server, thereby reducing the attack surface and enhancing overall security.

By combining the use of non-standard ports with fail2ban and UFW, the security posture of the remote server is significantly bolstered against unauthorized access attempts and potential malicious activity. These measures collectively contribute to mitigating the risks associated with operating services accessible over the Internet, safeguarding the confidentiality, integrity, and availability of the hosted resources.

1.10.1 Flask Routes - User Interface

The Flask application routes are defined to handle various URL endpoints and user interactions. Each route corresponds to a specific functionality within the web application, facilitating dynamic content generation and user engagement. The routes are structured to provide a seamless user experience, enabling users to interact with the hydroponics system, view sensor data, and access community-driven features. By defining custom routes, developers can create a tailored web application that meets the specific requirements of the hydroponics project. In simple terms, a route is a URL pattern associated with a specific function in the Flask application. When a user navigates to a particular URL, the corresponding function is executed, generating the content displayed on the web page.

At the start, the routes were set up to only serve static HTML pages, not templates. This was done to ensure that the basic functionality of the server was working. Once the basic functionality was confirmed, the routes were updated to serve templates. The templates were created using HTML, CSS, Javascript, and Jinja2 templating engine. The Jinja2 templating engine allows for the dynamic generation of HTML content based on variables passed from the Flask application. This enables the creation of interactive web pages that can display real-time sensor data, user inputs, and other dynamic content.

The Flask application encompasses several web pages and routes, each designed to fulfill specific functionalities within the hydroponics management system. The homepage serves as the landing page of the application and acts as the entry point for users, providing links to login, account creation, and information about the hydroponics system. This page is delivered directly via Flask as a static HTML file.

The login route supports both GET and POST requests; GET serves the login form while POST processes the login data. It checks credentials against the credentials collection in MongoDB, utilizes Werkzeug for password verification, and manages session data for logged-in users. Similarly, the create account route manages the registration of new users and supports GET for serving the registration form and POST for processing and storing user data. It ensures usernames are unique, hashes passwords for security, and initializes user-specific entries in other MongoDB collections like settings, data, and messages.

The settings page allows users to adjust settings related to their hydroponics system through a Jinja2 template that dynamically generates a form based on current settings stored in MongoDB. Users can adjust parameters such as watering frequency and light duration through sliders, and form submission updates the user-specific settings in the database.

The friends page displays a list of friends or other users from which settings can be downloaded. It fetches friend information from the credentials collection and renders it using a Jinja2 template. Each friend entry includes a button to download settings, triggering a fetch request to a specific route that handles settings retrieval.

The forum page serves as a community platform where users can post discussions or read posts, backed by MongoDB to store posts. It dynamically renders content through a Jinja2 template, allowing users to interact by adding new posts or viewing existing discussions. The plants page displays real-time data from various sensors related to the user's plants, fetching and displaying sensor data stored in the data collection of MongoDB. The page updates dynamically to reflect real-time conditions of the hydroponics environment.

Additional functional routes include a download settings route that allows users to download and apply another user's settings to their system and a save settings route that receives POST requests from the settings page to update user-specific settings in MongoDB. Each route and page within the Flask application is carefully crafted to interact seamlessly with the MongoDB backend, ensuring data consistency and providing a responsive user experience. The use of Jinja2 templates enables dynamic content generation, making the web interface adaptable to real-time changes and user interactions.

Endpoint	**Supported Methods**
/	GET
/create_account	GET, POST
/login	GET, POST
/logout	GET
/plants	GET
/forum	GET
/add_post	POST
/friends	GET
/save_settings	POST
/settings	GET
/download_settings/<friend_id>	GET
/control_panel	GET

Table 2: Available Website Endpoints

1.10.2 Flask Routes - Application Programming Interface

The Raspberry Pi Pico W microcontroller communicates with the server through an action-driven API, enabling data exchange and system control. The API routes are designed to handle requests from the microcontroller, process sensor data, and update the database with real-time information. By defining custom API routes, a structured communication protocol between the microcontroller and the server was established, facilitating seamless data transmission and system control.

The API is characterized as a functional API or an action-based API, distinct from a strictly RESTful architecture. Its design revolves around executing specific actions and accessing targeted functionalities within the application, encompassing tasks such as retrieving sensor data, updating settings, managing hardware IDs, and controlling manual overrides. Each API endpoint corresponds to a particular operation, such as retrieving settings or updating manual overrides, and ensures stateless communication, with every request containing sufficient information for independent processing by the server. JSON payloads are consistently structured in responses, complemented by appropriate HTTP status codes signaling request outcomes. The API's design prioritizes scalability and flexibility, enabling clients to interact with discrete functionalities without rigid coupling to underlying data structures or database schemas, thereby accommodating diverse application requirements.

Endpoint	Supported Methods
/api/test	GET
/api/get_hardware_id	GET
/api/set_hardware_id	GET
/api/update_settings	GET
/api/get_settings	GET
/api/get_humidity_data	GET
/api/get_temp_data	GET
/api/get_uv_data	GET
/api/get_ph_data	GET
/api/get_tank_data	GET
/api/get_all_data	GET
/api/get_manual_override	GET
/api/update_manual_override	POST
/api/update_uv_data	POST
/api/update_ph_data	POST
/api/update_temp_data	POST
/api/update_humidity_data	POST
/api/update_tank_data	POST
/api/update_all_sensors_data	POST

Table 3: Available API Endpoints

1.10.3 MongoDB Database

The backend database used for the hydroponics system is MongoDB. MongoDB is a "NoSQL" database that stores data in flexible, JSON documents. It is a popular choice for web applications due to its scalability, flexibility, and ease of use. MongoDB's document-based data model allows for the storage of complex data structures, making it well-suited for applications with dynamic data requirements. The database is hosted on a remote server and accessed by the Flask application to store and retrieve user data, sensor readings, and system settings. MongoDB provides a robust and reliable data storage solution for the hydroponics system, enabling efficient data management and seamless integration with the web application.

There were four collection within the database: credentials (34a), data, settings, and messages (34b). The credentials collection stores user account information, including usernames, hashed passwords, first names, last names, and the paths to profile pictures. The data collection stores sensor readings and system data, such as temperature, humidity, pH, and water level. The settings collection stores user-specific settings, such as watering frequency, light duration, and nutrient levels. The messages collection stores user messages and forum posts, enabling community-driven interactions and discussions. Each collection is designed to store specific types of data and facilitate efficient data retrieval and management within the hydroponics system.

```
_id: ObjectId('662dea38974511e984f14e22')
first_name : "Liam"
last_name : "Goss"
username : "liamgoss"
password : "scrypt:32768:8:1$p7JoJVO3Yhs4Rdqj$bc4dfe5429de8a83af7b1606d0f04c00dc83…"
created : 2024-04-27T23:18:32.885+00:00
▾ friends : Array (3)
  ▾ 0: Object
      username : "luigivilla"
      user_id : "662dea46974511e984f14e26"
  ▾ 1: Object
      username : "abe123"
      user_id : "662dea5f974511e984f14e2a"
  ▾ 2: Object
      username : "ericriv"
      user_id : "662dea69974511e984f14e2e"
last_login : 2024-04-29T11:37:45.647+00:00
device_id : "e66164084383262d"
```


(a) Credentials Collection Sample Entry


```
_id: ObjectId('662fed2b32510c74817af0d7')
user_id : "662dea38974511e984f14e22"
name : "Liam"
message : "Can't wait to get started with LuLi"
created : 2024-04-29T11:55:39.774+00:00
image_path : "USER_IMAGES/fc7cfb3a-d2c0-424f-9a35-e82622d622e4.jpg"
```


(b) Messages Collection Sample Entry

Figure 34: MongoDB Credentials and Messages Collections

2 Future Work

The most desirable improvement that could be made is the creation of a single PCB for all components in lieu of a small PCB and a breadboard at the same time. This would mean increased surface area on the PCB would could be used to include two more pumps. One pump would assist the drainage of water in the plant containers while the other pump could be used to mix a nutrient solution into the water tank. While the size of the PCB increases, a smaller physical enclosure could potentially attract a wider audience and allow for easier placement in a home. The addition of a nutrient pump would allow for the system to be more self-sufficient and require less maintenance.

Additionally, the website does not adhere to privacy laws such as GDPR. The website should be updated to include a privacy policy and terms of service. The website should also include a cookie consent banner to comply with the EU's cookie law. The website should also include a way for users to delete their account and all associated data. This would require the addition of a delete account route and the deletion of all associated data from the database. Also, for demonstration purposes, when a new user joins the website, they are automatically added to the friends list of the every other user. This was intended to showcase the forum and friends functionalities. This should be removed in a production environment and instead users should be able to add friends manually.

Finally, a .com domain should be purchased for the website. This would make the website more professional and easier to remember. The website should also be hosted on a cloud server such as AWS or Google Cloud. This would allow for more reliable uptime and faster load times. The website currently uses a self-signed SSL certificate. This should be replaced with a valid SSL certificate from a trusted certificate authority. This would ensure that all data transmitted between the user and the server is encrypted and secure.

3 Conclusion

he Luli Hydroponics System offers an innovative and socially connected platform for soilless plant cultivation, integrating a variety of sensors and a microcontroller with a user-friendly web application. Central to its operation is the Raspberry Pi Pico W microcontroller, which orchestrates the environment monitoring by gathering real-time data on critical parameters such as pH, temperature, humidity, UV light, and water level. This allows for precise control and optimization of plant growth conditions. Users can access the LuLi web portal not only to monitor their system but also to connect with a wider community of urban horticulturalists, sharing insights and hydroponic data, thus creating a democratized hub for hydroponic enthusiasts. The system's modular architecture not only provides a robust physical environment but also facilitates future enhancements and customizations to meet diverse horticultural needs. Developed within a nine-month period on a budget of $500, the project showcases the effective application of engineering skills, with the team demonstrating efficient time management and collaborative problem-solving to meet the project timeline. The Luli Hydroponics System stands as a testament to the fusion of technology and social connectivity in modern urban agriculture.

A Software Appendix

A.1 Raspberry Pi Pico W MicroPython Code

A.1.1 Luli_main.py

```
1
2  from Luli_UV import LTR390
3  from Luli_Ultrasonic import WaterLevelSensor
4  from Luli_pH import PHSensor
5  from Luli_OLED import OLEDMenuDisplay
6  from Luli_DHT import DHT22Handler
7  from Luli_Networkhandler import NetworkHandler
8  from Luli_MotorLEDControl import MotorAndLEDControl
9  import machine
10 import Luli_CONFIG
11 import utime
12
13 MOTOR_LED_CONTROL = MotorAndLEDControl()
14
15 # Sensor Initialization
16 PH_SENSOR = PHSensor()
17 UV_SENSOR = LTR390()
18 TANK_SENSOR = WaterLevelSensor()
19 DHTS = DHT22Handler()
20
21
22 # External Hardware/Misc Initialization
23 OLED = OLEDMenuDisplay()
24 NETWORK = NetworkHandler(Luli_CONFIG.WIFI_SSID, Luli_CONFIG.
       WIFI_PASSWORD, Luli_CONFIG.SERVER_URL)
25
26 def log_error(e):
27     with open('error_log.txt', 'a') as file:  # 'a' opens the file
           for appending
28         timestamp = utime.localtime()  # Get the current time
29         readable_timestamp = "{year}/{month}/{day} {hours}:{
               minutes}:{seconds}".format(
30             year=timestamp[0], month=timestamp[1], day=timestamp
                   [2],
31             hours=timestamp[3], minutes=timestamp[4], seconds=
                   timestamp[5]
32         )
33         file.write('{} - Error: {}\n'.format(readable_timestamp, e
               ))
34
35 def sensor_uv_read():
36     # Give the sensor some time to start
37     utime.sleep(Luli_CONFIG.DELAY_UV_STARTUP)
38     raw, uv_intensity = UV_SENSOR.get_uv_data()
39     print("RAW: {}".format(raw))
```

```
40      print("UV Intensity: {}uW/cm^2".format(uv_intensity))
41      #return raw, uv_intensity
42      return uv_intensity
43
44  def sensor_ph_read():
45      ph_level = PH_SENSOR.get_ph()
46      print("Current pH level:", ph_level)
47      return ph_level
48
49  #def sensor_tank_calibrate():
50  #    TANK_SENSOR.calibrate_tank_full()
51  #    TANK_SENSOR.calibrate_tank_empty()
52
53  def sensor_tank_read():
54      return TANK_SENSOR.get_water_level()
55
56
57  def sensor_temp_read():
58      # Compute average temp of sensors 0 to 3
59      total_temp = 0
60      for i in range(4):
61          # Average and convert to Fahrenheit
62          temp = DHTS.get_temperature(i)
63          num_DHTs = 4
64          if temp is not None:
65              total_temp += ((temp * 9/5) + 32)
66          else:
67              total_temp += 0
68              num_DHTs = num_DHTs - 1
69      if num_DHTs <= 1:
70          num_DHTs = 1
71      avg_temp = total_temp / num_DHTs
72      print("Average temperature:", avg_temp)
73      return avg_temp
74
75  def sensor_humidity_read():
76      # Compute average humidity of sensors 0 to 3
77      total_hum = 0
78      num_DHTs = 4
79      for i in range(4):
80          hum = DHTS.get_humidity(i)
81          if hum is not None:
82              total_hum += DHTS.get_humidity(i)
83
84          else:
85              total_hum += 0
86              num_DHTs = num_DHTs - 1
87      if num_DHTs <= 1:
88          num_DHTs = 1
89      avg_hum = total_hum / num_DHTs
90      print("Average humidity:", avg_hum)
```

```
91      return avg_hum
92
93  def update_display(plant_name='Lettuce', planted_date='3/15/24',
        harvest_date='5/10/24', ph_param=None, temp_param=None,
        light_param=None, humidity_param=None, tank_param=None):
94      ph = ph_param if ph_param is not None else sensor_ph_read()
95      temp = temp_param if temp_param is not None else
            sensor_temp_read()
96      light = light_param if light_param is not None else
            sensor_uv_read()
97      humidity = humidity_param if humidity_param is not None else
            sensor_humidity_read()
98      tank = tank_param if tank_param is not None else
            sensor_tank_read()
99
100     OLED.print_sensor_data(ph=ph, temp=temp, light=light, humidity
            =humidity, tank=tank)
101     utime.sleep(Luli_CONFIG.DELAY_OLED)
102     OLED.print_plant_menu(plant_name=plant_name, planted_date=
            planted_date, harvest_date=harvest_date)
103     utime.sleep(Luli_CONFIG.DELAY_OLED)
104
105 def send_all_data(ph_param=None, temp_param=None, light_param=None
        , humidity_param=None, tank_param=None):
106     ph = ph_param if ph_param is not None else sensor_ph_read()
107     temp = temp_param if temp_param is not None else
            sensor_temp_read()
108     light = light_param if light_param is not None else
            sensor_uv_read()
109     humidity = humidity_param if humidity_param is not None else
            sensor_humidity_read()
110     tank = tank_param if tank_param is not None else
            sensor_tank_read()
111
112
113     data = {
114         "ph": ph,
115         "temp": temp,
116         "light": light,
117         "humidity": humidity,
118         "tank": tank
119     }
120     try:
121         response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_ALL_SENSOR_DATA, data)
122     except Exception as e:
123         print("Error sending all sensor data:", e)
124         #log_error(e)
125     return response
126
127 def send_data(data, data_label):
```

```
128     response = None
129
130     if data_label == 'ph':
131         try:
132             response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_PH_DATA, data)
133         except Exception as e:
134             print("Error sending pH data:", e)
135             #log_error(e)
136     elif data_label == 'temp':
137         try:
138             response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_TEMP_DATA, data)
139         except Exception as e:
140             print("Error sending temperature data:", e)
141             #log_error(e)
142     elif data_label == 'light':
143         try:
144             response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_UV_DATA, data)
145         except Exception as e:
146             print("Error sending UV data:", e)
147             #log_error(e)
148     elif data_label == 'humidity':
149         try:
150             response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_HUMIDITY_DATA, data)
151         except Exception as e:
152             print("Error sending humidity data:", e)
153             #log_error(e)
154     elif data_label == 'tank':
155         try:
156             response = NETWORK.send_data(Luli_CONFIG.
                ENDPOINT_UPDATE_TANK_DATA, data)
157         except Exception as e:
158             print("Error sending tank data:", e)
159             #log_error(e)
160     else:
161         print("Invalid data label")
162
163     # if response was successful, return True
164     if response.status_code == 200:
165         return True
166     else:
167         return False
168
169 def update_config(new_settings):
170     #Luli_CONFIG.UV_READ_INTERVAL = int(new_settings.get('
        uv_read_interval', Luli_CONFIG.UV_READ_INTERVAL))
171     #Luli_CONFIG.TEMP_HUMIDITY_READ_INTERVAL = int(new_settings.
        get('temp_humidity_read_interval', Luli_CONFIG.
```

```
        TEMP_HUMIDITY_READ_INTERVAL))
172     Luli_CONFIG.DURATION_WATER_CYCLE = int(new_settings.get('
        pump_duration', Luli_CONFIG.DURATION_WATER_CYCLE)) or
        Luli_CONFIG.DURATION_WATER_CYCLE
173     Luli_CONFIG.NEXT_WATER_CYCLE = int(new_settings.get('
        water_internval', Luli_CONFIG.NEXT_WATER_CYCLE)) or
        Luli_CONFIG.NEXT_WATER_CYCLE
174     Luli_CONFIG.DURATION_LED_CYCLE = int(new_settings.get('
        led_duration', Luli_CONFIG.DURATION_LED_CYCLE)) or
        Luli_CONFIG.DURATION_LED_CYCLE
175     print("Updated settings from server")
176
177
178 if __name__ == '__main__':
179     # Projects Day Demo Timings
180     # 1. Read UV Sensor Data every 2 minutes
181     # 2. Read pH Sensor Data and tank level data after every water
            pump cycle
182     # 3. Read Temperature and Humidity every 30 seconds
183
184
185     MOTOR_LED_CONTROL.motor_off()
186     MOTOR_LED_CONTROL.leds_on()
187
188     try:
189         # Start time
190         start_time = utime.time()
191         next_uv_time = start_time + Luli_CONFIG.UV_READ_INTERVAL
192         next_temp_humidity_time = start_time + Luli_CONFIG.
            TEMP_HUMIDITY_READ_INTERVAL
193         next_motor_start_time = start_time # Start motor on
            startup
194         next_motor_stop_time = start_time + Luli_CONFIG.
            DURATION_WATER_CYCLE
195         next_manual_override_check_time = start_time + Luli_CONFIG
            .MANUAL_OVERRIDE_CHECK_INTERVAL  # Setting up the next
            time to check for manual overrides
196         next_settings_update_time = start_time + Luli_CONFIG.
            SETTINGS_UPDATE_INTERVAL
197
198         temperature = None
199         humidity = None
200         ph = None
201         tank = None
202         uv = None
203
204         # Assumes tank is full on startup
205         TANK_SENSOR.calibrate_tank_full()
206
207         while True:
208
```

```
209             current_time = utime.time()
210
211
212             # Start Water Cycle that reads pH and tank level
213             if current_time >= next_motor_start_time:
214                 tank_level_percent = sensor_tank_read()  # Get the
                        current water level percentage
215                 if tank_level_percent is not None and
                        tank_level_percent > Luli_CONFIG.
                        MINIMUM_WATER_LEVEL:
216                     MOTOR_LED_CONTROL.motor_on()
217                     next_motor_stop_time = current_time +
                            Luli_CONFIG.DURATION_WATER_CYCLE  #
                            Schedule to stop
218                     next_motor_start_time = next_motor_stop_time +
                             Luli_CONFIG.NEXT_WATER_CYCLE
219                 else:
220                     print("Water level too low to start pump")
221                     #next_motor_start_time = current_time +
                            Luli_CONFIG.CHECK_WATER_LEVEL_INTERVAL  #
                            Check again after some time
222
223             #print("CURRENT: ", current_time)
224             #print("SToP TIME: ", next_motor_stop_time)
225             if current_time >= next_motor_stop_time:
226                 MOTOR_LED_CONTROL.motor_off()
227                 print("MOTOR SHOULD BE OFF")
228                 ph = sensor_ph_read()
229                 tank = sensor_tank_read()  # Read again after pump
                        off in case level has changed
230                 update_display(ph_param=ph, tank_param=tank)
231                 try:
232                     send_data(ph, 'ph')
233                     send_data(tank, 'tank')
234                 except Exception as e:
235                     print("Error sending pH/tank data:", e)
236                 next_motor_start_time = current_time + Luli_CONFIG
                        .NEXT_WATER_CYCLE  # Schedule next cycle
237
238             # Check for manual override every 30 seconds
239             if current_time >= next_manual_override_check_time:
240                 try:
241                     # Check for manual override
242                     override_commands = NETWORK.
                            get_manual_override()
243                     if override_commands:
244                         print("Manual override commands received:"
                                , override_commands)
245                         if 'motor' in override_commands:
246                             if override_commands['motor'] == 'on':
247                                 MOTOR_LED_CONTROL.motor_on()
```

```
                        elif override_commands['motor'] == 'off':
                            MOTOR_LED_CONTROL.motor_off()
                    if 'leds' in override_commands:
                        if override_commands['leds'] == 'on':
                            MOTOR_LED_CONTROL.leds_on()
                        elif override_commands['leds'] == 'off':
                            MOTOR_LED_CONTROL.leds_off()
                    if 'read_light' in override_commands:
                        uv = sensor_uv_read()
                        update_display()
                        send_data(uv, 'light')
                    if 'read_temp' in override_commands:
                        temperature = sensor_temp_read()
                        update_display()
                        send_data(temperature, 'temp')
                    if 'read_humidity' in override_commands:
                        humidity = sensor_humidity_read()
                        update_display()
                        send_data(humidity, 'humidity')
                    if 'read_ph' in override_commands:
                        ph = sensor_ph_read()
                        update_display(ph_param=ph)
                        send_data(ph, 'ph')
                    if 'read_tank' in override_commands:
                        tank = sensor_tank_read()
                        update_display(tank_param=tank)
                        send_data(tank, 'tank')
                except Exception as e:
                    print("Error getting manual override:", e)
                    #log_error(e)

                # Update the next check time for manual overrides
                next_manual_override_check_time = current_time + Luli_CONFIG.MANUAL_OVERRIDE_CHECK_INTERVAL

            # Check for settings update every 2 minutes
            if current_time >= next_settings_update_time:
                try:
                    new_settings = NETWORK.fetch_and_update_settings()
                    if new_settings:
                        update_config(new_settings)
                    next_settings_update_time = current_time + Luli_CONFIG.SETTINGS_UPDATE_INTERVAL
                except Exception as e:
                    print("Error fetching settings:", e)
                    #log_error(e)

            # Read UV and update display every 120 seconds
```

```
            if current_time >= next_uv_time:
                uv = sensor_uv_read()
                update_display()
                try:
                    send_data(uv, 'light')
                except Exception as e:
                    print("Error sending UV data:", e)
                next_uv_time = current_time + Luli_CONFIG.
                   UV_READ_INTERVAL  # Schedule next run

            # Read Temperature and Humidity every 30 seconds
            if current_time >= next_temp_humidity_time:
                temperature = sensor_temp_read()
                humidity = sensor_humidity_read()
                update_display()
                try:
                    send_data(temperature, 'temp')
                    send_data(humidity, 'humidity')
                except Exception as e:
                    print("Error sending temp/humidity data:", e)
                next_temp_humidity_time = current_time +
                   Luli_CONFIG.TEMP_HUMIDITY_READ_INTERVAL #
                   Schedule next run




            utime.sleep(0.1)  # Sleep for 100ms to reduce CPU
               usage

    except Exception as e:
        # SHUTDOWN ERROR CODE
        # Log the error
        log_error(e)
        print("Error:", e)
        MOTOR_LED_CONTROL.motor_off()
        for _ in range(3):
            MOTOR_LED_CONTROL.leds_off()
            utime.sleep(0.5)
            MOTOR_LED_CONTROL.leds_on()
            utime.sleep(0.5)
        MOTOR_LED_CONTROL.leds_off()
        machine.reset() # Reset the board if an error occurs
```

Listing 1: Main MicroPython Code

A.1.2 Luli_CONFIG.py

```
# Luli Hydroponics Config File
# This file will define all necessary pins and timing constants
   for the Luli Hydroponics system
```

```
3
4  ######################
5  #    PIN SETTINGS    #
6  ######################
7  PIN_UV_SENSOR_SDA = 0
8  PIN_UV_SENSOR_SCL = 1
9
10 PIN_OLED_SCK = 10
11 PIN_OLED_MOSI = 11
12 PIN_OLED_CS = 13
13 PIN_OLED_RST = 14
14 PIN_OLED_DC = 15
15
16 PIN_ULTRASONIC_ECHO = 9
17 PIN_ULTRASONIC_TRIG = 17
18
19 PIN_DHT0 = 16
20 PIN_DHT1 = 18
21 PIN_DHT2 = 19
22 PIN_DHT3 = 20
23
24 PIN_MOTOR = 21
25 PIN_LED = 22
26 PIN_PH_SENSOR = 26
27
28
29
30
31 ##########################
32 #    TIMING CONSTANTS    #
33 ##########################
34 DELAY_OLED = 5 # seconds
35 DELAY_PH_RESPONSE = 50 # ms
36 DELAY_UV_STARTUP = 1 # seconds
37
38 TEMP_HUMIDITY_READ_INTERVAL = 30 # seconds
39 UV_READ_INTERVAL = 120 # seconds
40 MANUAL_OVERRIDE_CHECK_INTERVAL = 30 # seconds
41 SETTINGS_UPDATE_INTERVAL = 120 # seconds
42 DURATION_WATER_CYCLE = 60 # seconds
43 NEXT_WATER_CYCLE = 3600 # seconds
44 DURATION_LED_CYCLE = 3600 # seconds
45
46
47
48 ##########################
49 #    NETWORK SETTINGS    #
50 ##########################
51 WIFI_SSID = 'WIFI_NAME'
52 WIFI_PASSWORD = 'PASSWORD'
53 SERVER_URL = 'https://luli-hydroponics.ddns.net'
```

```
54
55
56
57 ##########################
58 #   ENDPOINT SETTINGS    #
59 ##########################
60 ENDPOINT_TEST = '/api/test'
61 ENDPOINT_GET_SETTINGS = '/api/get_settings'
62 ENDPOINT_SET_HARDWARE_ID = '/api/set_hardware_id'
63 ENDPOINT_UPDATE_SETTINGS = '/api/update_settings'
64 ENDPOINT_UPDATE_UV_DATA = '/api/update_uv_data'
65 ENDPOINT_UPDATE_PH_DATA = '/api/update_ph_data'
66 ENDPOINT_UPDATE_TEMP_DATA = '/api/update_temp_data'
67 ENDPOINT_UPDATE_HUMIDITY_DATA = '/api/update_humidity_data'
68 ENDPOINT_UPDATE_TANK_DATA = '/api/update_tank_data'
69 ENDPOINT_UPDATE_ALL_SENSOR_DATA = '/api/update_all_sensor_data'
70 ENDPOINT_MANUAL_OVERRIDE = '/api/get/manual_override'
71
72
73 ######################
74 #   MISC CONSTANTS   #
75 ######################
76 # https://optoelectronics.liteon.com/upload/download/DS86
      -2015-0004/LTR-390UV_Final_%20DS_V1%201.pdf
77 UVS_CONVERSION_FACTOR = 2300 # Conversion factor from the
      datasheet
78 VOLTAGE_AT_PH7 = 2.51
79 VOLTAGE_AT_PH4 = 3.03
80 MINIMUM_WATER_LEVEL = 20 # Minimum water level in the tank in
      percentage (%)
81 TANK_EMPTY_DISTANCE = 12 # Distance in cm when the tank is empty
```

Listing 2: Pico W Configuration File

A.1.3 Luli_DHT.py

```
1     import dht
2     from machine import Pin
3     import time
4     import Luli_CONFIG
5
6     class DHT22Handler:
7         def __init__(self):
8             # Initialize all DHT22 sensors on their respective
                 pins
9             self.sensors = [
10                dht.DHT22(Pin(Luli_CONFIG.PIN_DHT0, Pin.IN, Pin.
                     PULL_DOWN)),
11                dht.DHT22(Pin(Luli_CONFIG.PIN_DHT1, Pin.IN, Pin.
                     PULL_DOWN)),
```

```
12                  dht.DHT22(Pin(Luli_CONFIG.PIN_DHT2, Pin.IN, Pin.
                        PULL_DOWN)),
13                  dht.DHT22(Pin(Luli_CONFIG.PIN_DHT3, Pin.IN, Pin.
                        PULL_DOWN))
14              ]
15
16          def measure_all(self):
17              # Attempt to measure from all sensors up to three
                    times
18              retry = 0
19              while retry < 3:
20                  try:
21                      for sensor in self.sensors:
22                          sensor.measure()
23                      return True
24                  except Exception as e:
25                      print(".", end="")
26                      retry += 1
27                      time.sleep(0.5)
28              return False
29
30          def get_temperature(self, sensor_index):
31              # Return the temperature from the specified sensor
32              if self.measure_all():
33                  return self.sensors[sensor_index].temperature()
34              else:
35                  print("Failed to read temperature after several
                        attempts.")
36                  return None
37
38          def get_humidity(self, sensor_index):
39              # Return the humidity from the specified sensor
40              if self.measure_all():
41                  return self.sensors[sensor_index].humidity()
42              else:
43                  print("Failed to read humidity after several
                        attempts.")
44                  return None
45
46      def main():
47          dht_handler = DHT22Handler()
48          while True:
49              print("Measuring.")
50              for i in range(4):
51                  temp = dht_handler.get_temperature(i)
52                  hum = dht_handler.get_humidity(i)
53                  if temp is not None and hum is not None:
54                      print(f"Sensor {i}: Temperature: {temp} C,
                            Humidity: {hum} %")
55              time.sleep(3)
56
```

```
57      if __name__ == "__main__":
58          main()
```

Listing 3: Pico W DHT22 Sensor Code

A.1.4 Luli_MotorLEDControl.py

```
1       from machine import Pin
2       import time
3       import Luli_CONFIG
4
5       class MotorAndLEDControl:
6           def __init__(self, motor_pin=Luli_CONFIG.PIN_MOTOR,
                led_pin=Luli_CONFIG.PIN_LED):
7               # Initialize motor and LED pins
8               self.motor_gate = Pin(motor_pin, Pin.OUT)
9               self.led_gate = Pin(led_pin, Pin.OUT)
10
11          def motor_on(self):
12              # Turn the motor on
13              print("MOTOR ON")
14              self.motor_gate.value(1)
15
16          def motor_off(self):
17              # Turn the motor off
18              print("MOTOR OFF")
19              self.motor_gate.value(0)
20
21          def leds_on(self):
22              # Turn the LED on
23              print("LEDS ON")
24              self.led_gate.value(1)
25
26          def leds_off(self):
27              # Turn the LED off
28              print("LEDS OFF")
29              self.led_gate.value(0)
30
31      # Testing the class functionality
32      if __name__ == "__main__":
33          control = MotorAndLEDControl()
34
35          # Testing Motor Control
36          control.motor_off()
37          control.leds_off()
38          time.sleep(3)
39          control.motor_on()
40          time.sleep(45)
41          control.motor_off()
42          time.sleep(5)
43
```

```
        # Testing LED Control
        while True:
            control.led_on()
            time.sleep(1)
            control.led_off()
            time.sleep(1)
```

Listing 4: Pico W Low Side Switching MOSFET Control Code

A.1.5 Luli_Networkhandler.py

```
    import network
    import urequests as requests
    import ujson
    import machine
    import Luli_CONFIG
    class NetworkHandler:
        def get_hardware_id(self):
            # Get the unique hardware ID
            unique_id = machine.unique_id()
            # Convert ID bytes to hexadecimal string
            hex_string = "".join("{:02x}".format(byte) for byte in
                unique_id)
            print("Unique Hardware ID:", hex_string)
            return unique_id



        def __init__(self, ssid, password, base_url):
            self.ssid = Luli_CONFIG.WIFI_SSID
            self.password = Luli_CONFIG.WIFI_PASSWORD
            self.base_url = Luli_CONFIG.SERVER_URL
            self.wlan = network.WLAN(network.STA_IF)
            self.hardware_id = str(self.get_hardware_id())

        def connect_wifi(self):
            """Connects to WiFi using the SSID and password
               provided during initialization."""
            self.wlan.active(True)
            if not self.wlan.isconnected():
                print('Connecting to network...')
                self.wlan.connect(self.ssid, self.password)
                while not self.wlan.isconnected():
                    pass
            print('Network config:', self.wlan.ifconfig())

        def send_data(self, endpoint, data):
            """Sends data to a specified API endpoint using HTTP
               POST."""
            headers = {'Content-Type': 'application/json', 'X-
               Device-ID': self.hardware_id}
```

```
37              try:
38                  response = requests.post(self.base_url + endpoint,
                        data=ujson.dumps(data), headers=headers)
39                  print(response.text)
40                  # Free up memory once the request is complete
41                  if 'response' in locals():
42                      response.close()
43                  return response
44              except Exception as e:
45                  print("Failed to send data:", e)
46                  return None
47
48
49          def get_manual_override(self):
50              """Checks for manual override commands from the server
                    using device-specific endpoint."""
51              device_specific_endpoint = Luli_CONFIG.
                   ENDPOINT_MANUAL_OVERRIDE + '/' + self.hardware_id
52              try:
53                  response = requests.get(self.base_url +
                       device_specific_endpoint)
54                  if response.status_code == 200:
55                      commands = response.json()
56                      return commands
57                  else:
58                      print("Failed to get manual override, status
                           code:", response.status_code)
59                      return None
60              except Exception as e:
61                  print("Failed to get manual override:", e)
62                  return None
63              finally:
64                  if 'response' in locals():
65                      response.close()
66
67          def fetch_and_update_settings(self):
68              headers = {'X-Device-ID': Luli_CONFIG.DEVICE_ID}
69              try:
70                  response = requests.get(self.base_url +
                       Luli_CONFIG.ENDPOINT_GET_SETTINGS, headers=
                       headers)
71                  if response.status_code == 200:
72                      new_settings = response.json()
73                      return new_settings
74                  else:
75                      print("Failed to fetch settings:", response.
                           text)
76              except Exception as e:
77                  print("Failed to fetch settings:", e)
78                  return None
79              finally:
```

```
                if 'response' in locals():
                    response.close()


if __name__ == '__main__':
    ssid = 'YourSSID'
    password = 'YourPassword'
    base_url = 'http://yourapi.com'
    endpoint = '/api/update_all_sensor_data'
    sensor_data = {
        "temp": 23.5,
        "humidity": 45,
        "light": 300,
        "ph": 7.0,
        "tank": 80
    }

    network_handler = NetworkHandler(ssid, password, base_url)
    network_handler.connect_wifi()
    network_handler.send_data(endpoint, sensor_data)
```

Listing 5: Pico W Network Handler Code

A.1.6 Luli_OLED.py

```
from machine import Pin, SPI
import ssd1306
import time
import Luli_CONFIG

class OLEDMenuDisplay:
    def __init__(self):
        # SPI setup
        self.spi = SPI(1, baudrate=1000000, polarity=1, phase=1, bits=8, firstbit=SPI.MSB,
                       sck=Pin(Luli_CONFIG.PIN_OLED_SCK), mosi=Pin(Luli_CONFIG.PIN_OLED_MOSI))
        # Display setup
        self.cs = Pin(Luli_CONFIG.PIN_OLED_CS)   # Chip select
        self.dc = Pin(Luli_CONFIG.PIN_OLED_DC)   # Data/command
        self.rst = Pin(Luli_CONFIG.PIN_OLED_RST, Pin.OUT)   # Reset
        self.display = ssd1306.SSD1306_SPI(128, 64, self.spi, self.dc, self.rst, self.cs)

        self.display.poweron()
        self.display.show()

    def init_main_menu(self):
```

```
21              self.display.fill(0)
22              self.display.rect(0, 0, 10, 20, 1)
23              self.display.rect(11, 0, 117, 20, 1)
24              self.display.text('Sensor Data', 13, 5, 1)
25
26              self.display.rect(0, 21, 10, 20, 1)
27              self.display.rect(11, 21, 117, 20, 1)
28              self.display.text('Cycle Info', 13, 25, 1)
29
30              self.display.rect(0, 42, 10, 20, 1)
31              self.display.rect(11, 42, 117, 20, 1)
32              self.display.text('Plants', 13, 45, 1)
33              self.display.show()
34
35          def print_main_menu(self, select):
36              self.display.fill(0)
37              menu_highlight = 20 * select + select
38              self.init_main_menu()
39              self.display.fill_rect(0, menu_highlight, 10, 20, 1)
40              self.display.show()
41
42          def print_sensor_data(self, ph, temp, light, humidity,
                tank):
43              self.display.fill(0)
44              self.display.text('Tank: ' + str(tank), 0, 0, 1)
45              self.display.text('pH: ' + str(ph), 0, 16, 1)
46              self.display.text('Temp: ' + str(temp), 0, 27, 1)
47              self.display.text('Light: ' + str(light), 0, 39, 1)
48              self.display.text('Humidity: ' + str(humidity), 0, 51,
                    1)
49              self.display.show()
50
51          def print_plant_menu(self, plant_name='Spinach',
                planted_date='5/5/24', harvest_date='9/3/24'):
52              self.display.fill(0)
53              self.display.text('Plants', 0, 0, 1)
54              self.display.text(f'Plant: {plant_name}', 0, 16, 1)
55              self.display.text(f'Planted: {planted_date}', 0, 27,
                    1)
56              self.display.text(f'Harvest: {harvest_date}', 0, 39,
                    1)
57              self.display.show()
58
59      def main():
60          display = OLEDMenuDisplay()
61          #display.init_main_menu()
62          time.sleep(1)
63
64          # Loop through menu printing functions as an example
65          while True:
66              display.print_sensor_data(7.2, 22.5, 150, 45, 'Full')
```

```
            time.sleep(6)
            display.print_plant_menu('Tomatoes', '4/1/24', '
               6/28/24')
            time.sleep(6)

    if __name__ == "__main__":
        main()
```

Listing 6: Pico W OLED Display Code

A.1.7 Luli_pH.py

```
    from machine import ADC, Pin
    import utime
    import Luli_CONFIG

    class PHSensor:
        # Constants for pH calculation
        VOLTAGE_AT_PH7 = Luli_CONFIG.VOLTAGE_AT_PH7  # Voltage at
           pH 7
        VOLTAGE_AT_PH4 = Luli_CONFIG.VOLTAGE_AT_PH4  # Voltage at
           pH 4
        PH_STEP = (VOLTAGE_AT_PH7 - VOLTAGE_AT_PH4) / (7 - 4)  #
           Voltage change per pH unit

        def __init__(self, pin_number=Luli_CONFIG.PIN_PH_SENSOR):
            self.adc = ADC(Pin(pin_number))  # Initialize ADC on
               specified pin

        def read_sensor(self):
            buf = []
            for _ in range(10):
                adc_value = self.adc.read_u16()
                buf.append(adc_value)
                utime.sleep_ms(Luli_CONFIG.DELAY_PH_RESPONSE)  #
                   Sleep to match sensor response time
            buf.sort()
            avg_adc_value = sum(buf[1:-1]) / (len(buf) - 2)  #
               Calculate average excluding outliers
            return avg_adc_value

        def adc_to_ph(self, adc_value):
            voltage = adc_value * (3.3 / 65535)  # Convert ADC
               value to voltage
            ph_value = 7 + (self.VOLTAGE_AT_PH7 - voltage) / self.
               PH_STEP  # Calculate pH
            return ph_value

        def get_ph(self):
            avg_adc_value = self.read_sensor()
            return self.adc_to_ph(avg_adc_value)
```

```
32
33      def main():
34          ph_sensor = PHSensor()
35          while True:
36              ph = ph_sensor.get_ph()
37              print("Sensor pH level =", ph)
38              utime.sleep(2)
39
40      if __name__ == "__main__":
41          main()
```

Listing 7: Pico W pH Sensor Code

A.1.8 Luli_Ultrasonic.py

```
1       import machine
2       import utime
3       import Luli_CONFIG
4
5       class WaterLevelSensor:
6           def __init__(self, trig_pin=Luli_CONFIG.
               PIN_ULTRASONIC_TRIG, echo_pin=Luli_CONFIG.
               PIN_ULTRASONIC_ECHO):
7               self.trig = machine.Pin(trig_pin, machine.Pin.OUT)
8               self.echo = machine.Pin(echo_pin, machine.Pin.IN)
9               self.tank_full_distance = None
10              self.tank_empty_distance = None
11
12          def read_distance(self):
13              self.trig.low()
14              utime.sleep_us(5)
15
16              self.trig.high()
17              utime.sleep_us(10)
18              self.trig.low()
19
20              timeout = 10000  # Timeout in microseconds
21
22              start_time = utime.ticks_us()
23              while self.echo.value() == 0:
24                  if utime.ticks_diff(utime.ticks_us(), start_time)
                       > timeout:
25                      return None  # Timeout with no echo received
26
27              signal_off = utime.ticks_us()
28
29              start_time = utime.ticks_us()
30              while self.echo.value() == 1:
31                  if utime.ticks_diff(utime.ticks_us(), start_time)
                       > timeout:
```

```
32                  return None  # Timeout while echo is still
                        high
33
34              signal_on = utime.ticks_us()
35
36              time_passed = utime.ticks_diff(signal_on, signal_off)
37              distance = (time_passed * 0.0343) / 2  # Convert time
                    to distance
38              return distance
39
40          def calibrate_tank_full(self):
41              print("Calibrating tank full... Please ensure the tank
                    is full.")
42              utime.sleep(5)
43              self.tank_full_distance = self.read_distance()
44              if self.tank_full_distance is not None:
45                  print("Tank full distance calibrated at:", self.
                        tank_full_distance, "cm")
46              else:
47                  print("Calibration failed. Make sure the tank is
                        full and try again.")
48
49          def calibrate_tank_empty(self):
50              print("Calibrating tank empty... Please ensure the
                    tank is empty.")
51              utime.sleep(5)
52              self.tank_empty_distance = self.read_distance()
53              if self.tank_empty_distance is not None:
54                  print("Tank empty distance calibrated at:", self.
                        tank_empty_distance, "cm")
55              else:
56                  print("Calibration failed. Make sure the tank is
                        empty and try again.")
57
58          def display_water_level(self):
59              distance = self.read_distance()
60              if distance is not None and distance > 2:
61                  if self.tank_full_distance is not None and self.
                        tank_empty_distance is not None:
62                      water_level = self.tank_empty_distance -
                            distance
63                      tank_capacity = self.tank_empty_distance -
                            self.tank_full_distance
64                      percent_full = (water_level / tank_capacity) *
                             100
65                      print("Water level:", round(percent_full, 2),
                            "% full")
66                  else:
67                      print("Tank 'full' or 'empty' distance not
                            calibrated.")
68              else:
```

```
                print("Out of range, too close, or sensor error.")

        def get_water_level(self):
            distance = self.read_distance()
            if distance is not None and distance > 2:
                if self.tank_full_distance is not None and self.
                   tank_empty_distance is not None:
                    water_level = self.tank_empty_distance -
                       distance
                    tank_capacity = self.tank_empty_distance -
                       self.tank_full_distance
                    percent_full = (water_level / tank_capacity) *
                        100
                    return round(percent_full, 2)
                else:
                    return None
            else:
                return None

    def main():
        sensor = WaterLevelSensor()
        # Uncomment to calibrate on startup
        # sensor.calibrate_tank_full()
        # sensor.calibrate_tank_empty()

        while True:
            sensor.display_water_level()
            utime.sleep(1)

    if __name__ == "__main__":
        main()
```

Listing 8: Pico W Ultrasonic Sensor Code

A.1.9 Luli_UV.py

```
    # Adapted from  https://forums.pimoroni.com/t/ltr390-
       micropython-code/22314/2
    import utime
    from machine import Pin, I2C
    import Luli_CONFIG

    class LTR390:
        # Constants
        ADDR = 0x53
        MAIN_CTRL = 0x00
        MEAS_RATE = 0x04
        GAIN = 0x05
        PART_ID = 0x06
        MAIN_STATUS = 0x07
        ALSDATA = 0x0D
```

```
15          UVSDATA = 0x10
16          INT_CFG = 0x19
17          INT_PST = 0x1A
18          THRESH_UP = 0x21
19          THRESH_LOW = 0x24
20          RESOLUTION_20BIT_utime400MS = 0x00
21          RESOLUTION_19BIT_utime200MS = 0x10
22          RESOLUTION_18BIT_utime100MS = 0x20  # Default
23          RESOLUTION_17BIT_utime50MS = 0x03
24          RESOLUTION_16BIT_utime25MS = 0x40
25          RESOLUTION_13BIT_utime12_5MS = 0x50
26          RATE_25MS = 0x0
27          RATE_50MS = 0x1
28          RATE_100MS = 0x2  # Default
29          RATE_200MS = 0x3
30          RATE_500MS = 0x4
31          RATE_1000MS = 0x5
32          RATE_2000MS = 0x6
33          GAIN_1 = 0x0
34          GAIN_3 = 0x1  # Default
35          GAIN_6 = 0x2
36          GAIN_9 = 0x3
37          GAIN_18 = 0x4
38
39          def __init__(self):
40              self.i2c = I2C(0, scl=Pin(Luli_CONFIG.
                    PIN_UV_SENSOR_SCL), sda=Pin(Luli_CONFIG.
                    PIN_UV_SENSOR_SDA), freq=100000)
41              self.ID = self.read_byte(self.PART_ID)
42              if self.ID != 0xB2:
43                  print("Read ID error! Check the hardware...")
44                  return
45              self.write_byte(self.MAIN_CTRL, 0x0A)  # UVS in Active
                    Mode
46              self.write_byte(self.MEAS_RATE, self.
                    RESOLUTION_20BIT_utime400MS | self.RATE_2000MS)
47              self.write_byte(self.GAIN, self.GAIN_18)
48
49          def read_byte(self, cmd):
50              data = self.i2c.readfrom_mem(self.ADDR, cmd, 1)
51              return data[0]
52
53          def write_byte(self, cmd, val):
54              self.i2c.writeto_mem(self.ADDR, cmd, bytes([val]))
55
56          def read_uvs(self):
57              data1 = self.read_byte(self.UVSDATA)
58              data2 = self.read_byte(self.UVSDATA + 1)
59              data3 = self.read_byte(self.UVSDATA + 2)
60              return (data3 << 16) | (data2 << 8) | data1
61
```

```
62          def get_uv_intensity(self):
63              raw_uvs = self.read_uvs()
64
65              uv_intensity = (raw_uvs / Luli_CONFIG.
                    UVS_CONVERSION_FACTOR) * 1000
66              return uv_intensity
67
68          def get_uv_data(self):
69              raw_uvs = self.read_uvs()
70              uv_intensity = (raw_uvs / Luli_CONFIG.
                    UVS_CONVERSION_FACTOR) * 1000
71              return raw_uvs, uv_intensity
72
73
74
75      if __name__ == '__main__':
76          sensor = LTR390()
77          utime.sleep(1)
78          try:
79              while True:
80                  raw, uv_intensity = sensor.get_uv_data()
81                  print("RAW: {}".format(raw))
82                  print("UV intensity: {}".format(uv_intensity))
83                  #return raw, uv_intensity
84                  #return uv_intensity
85          except KeyboardInterrupt:
86              print("Interrupted by user")
87              exit()
```

Listing 9: Pico W UV Sensor Code

A.1.10 MUX_Test_Code.py

```
1       from machine import Pin, ADC
2       import time
3
4       # Setup the data pin
5       data_pin = ADC(Pin(28))  # Assuming GPIO0 is set up as ADC for
            voltage reading
6
7       # Setup select pins
8       sA = Pin(2, Pin.OUT)
9       sB = Pin(3, Pin.OUT)
10      sC = Pin(4, Pin.OUT)
11
12      def select_input(a, b, c):
13          """Set the select pins based on the desired input"""
14          sA.value(a)
15          sB.value(b)
16          sC.value(c)
17
```

```
    def read_voltage():
        """Read and return the voltage from the ADC connected to
           the MUX output"""
        voltage = data_pin.read_u16() * 3.3 / 65535  # Convert ADC
            reading to voltage (3.3V reference)
        return voltage


    while True:
        # Test the MUX by selecting different inputs and printing
           the read voltage
        for input_number in range(3):  # Test for D0, D1, and D2
            select_input((input_number & 0x4) >> 2, (input_number
               & 0x2) >> 1, input_number & 0x1)
            time.sleep(1)  # Wait a bit for MUX to stabilize and
               ADC to settle
            voltage = read_voltage()
            print(f"Input {input_number}: Voltage = {voltage:.2f}
               V")
```

Listing 10: Pico W MUX Test Code

A.2 Flask Backend Python Code

A.2.1 main.py

```
    ########################################
    #           Luli Hydroponics           #
    #      Backend Code made with Flask    #
    ########################################


    from flask import Flask, render_template, request, redirect,
       url_for, flash, session, make_response, jsonify
    import os
    from pymongo import MongoClient
    from bson.objectid import ObjectId
    from werkzeug.security import generate_password_hash,
       check_password_hash
    from werkzeug.utils import secure_filename
    from dotenv import load_dotenv
    from datetime import datetime
    import uuid

    """
    ROUTES and METHODS
    - '/' : GET
    - '/create_account' : GET, POST
    - '/login' : GET, POST
    - '/logout' : GET
    - '/plants' : GET
```

```
24      - '/forum' : GET
25      - '/add_post' : POST
26      - '/friends' : GET
27      - '/save_settings' : POST
28      - '/settings' : GET
29      - '/download_settings/<friend_id>' : GET
30      - '/control_panel' : GET
31
32      """
33
34
35      # Get the absolute path to the directory where the script is
            located
36      BASE_DIR = os.path.dirname(os.path.abspath(__file__))
37
38
39      # Construct the absolute paths to 'static' and 'templates'
            folders
40      absolute_static_path = os.path.join(BASE_DIR, '..', 'frontend'
            , 'static')
41      absolute_templates_path = os.path.join(BASE_DIR, '..', '
            frontend', 'templates')
42      absolute_user_images_path = os.path.join(BASE_DIR, '..', '
            frontend', 'static', 'USER_IMAGES')
43
44      # Normalize the paths to remove any relative path components (
            like '..')
45      absolute_static_path = os.path.normpath(absolute_static_path)
46      absolute_templates_path = os.path.normpath(
            absolute_templates_path)
47      absolute_user_images_path = os.path.normpath(
            absolute_user_images_path)
48
49
50      # MongoDB connection setup
51      mongodb_connection_string = 'mongodb://localhost:27017'
52      # brew services start mongodb/brew/mongodb-community
53
54      client = MongoClient(mongodb_connection_string)
55      DATABASE = client['user_data']
56      users_collection = DATABASE.credentials
57      users_messages_collection = DATABASE.messages
58      users_settings_collection = DATABASE.settings
59      users_data_collection = DATABASE.data
60
61      app = Flask(__name__,
62                  static_url_path='',
63                  static_folder=absolute_static_path,
64                  template_folder=absolute_templates_path )
65
66      load_dotenv() # Load .env file data
```

```
67      app.secret_key = os.getenv("luli_secret_key") # Load secret
            key into flask app
68      os.environ.pop('luli_secret_key', None) # Delete secret key
            from environment variables
69
70
71      app.config['SESSION_COOKIE_SECURE'] = True  # Only send
            cookies over HTTPS.
72      app.config['SESSION_COOKIE_HTTPONLY'] = True  # Prevent client
            -side JS from accessing the cookie.
73      app.config['IMAGE_UPLOAD_FOLDER'] = absolute_user_images_path
74      app.config['PROFILE_PIC_FOLDER'] = os.path.join(
            absolute_static_path, 'PROFILE_PICS')
75      app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024  # 16MB
            max size
76      # Ensure the directory exists
77      os.makedirs(app.config['IMAGE_UPLOAD_FOLDER'], exist_ok=True)
78      os.makedirs(app.config['PROFILE_PIC_FOLDER'], exist_ok=True)
79
80      #app.config['PERMANENT_SESSION_LIFETIME'] = 600  # Session
            lifetime in seconds or use timedelta for more precision.
81
82
83      # Import the function to register API routes after the app
            instance is created
84      from api import register_api_routes
85
86      # Call the register function with the app object to set up the
             api endpoints
87      register_api_routes(app, users_data_collection=
            users_data_collection, users_settings_collection=
            users_settings_collection, users_collection=
            users_collection)
88
89      ##################
90      #   HOMEPAGE     #
91      ##################
92      @app.route('/')
93      def serve_welcome_page():
94          return app.send_static_file('Luli.html')
95
96
97      ########################
98      #   CREATE ACCOUNT     #
99      ########################
100     @app.route('/create_account', methods=['GET', 'POST'])
101     def create_account():
102         if request.method == 'POST':
103             first_name = request.form['first-name']
104             last_name = request.form['last-name']
105             username = request.form['username']
```

```
106             password = request.form['password']
107             profile_pic = request.files['profile_pic']
108             # Check if the username already exists in the database
109             if users_collection.find_one({'username': username}):
110                 return "Username already exists. Choose a
                        different one."
111
112             # Hash the password for security
113             hashed_password = generate_password_hash(password)
114
115             # Insert new user into the database
116             current_time = datetime.now()
117             result = users_collection.insert_one({
118                 'first_name': first_name,
119                 'last_name': last_name,
120                 'username': username,
121                 'password': hashed_password,
122                 'created': current_time,
123                 'friends': []  # Initialize an empty list for
                        friends
124             })
125
126             # Retrieve the new user's ID
127             new_user_id = result.inserted_id
128
129             if profile_pic and allowed_file(profile_pic.filename):
130                 filename = secure_filename(profile_pic.filename)
131                 # Standardize the file extension
132                 if filename.lower().endswith('.jpeg'):
133                     filename = filename[:-5] + '.jpg'  # Change
                            extension from .jpeg to .jpg
134
135                 filename = username + '.jpg'  # Rename the file to
                        username.jpg
136                 profile_pic_path = os.path.join(app.config['
                    PROFILE_PIC_FOLDER'], filename)
137                 profile_pic.save(profile_pic_path)
138
139             # Retrieve all existing users to add to the new user's
                    friend list, excluding the new user themselves
140             all_users = users_collection.find({'_id': {'$ne':
                new_user_id}}, {'_id': 1, 'username': 1})
141             friends_list = [{'username': user['username'], '
                user_id': str(user['_id'])} for user in all_users]
142
143             # Add existing users to the new user's friend list
144             users_collection.update_one({'_id': new_user_id}, {'
                $set': {'friends': friends_list}})
145
146             # Also add this new user to existing users' friends
                    lists
```

```
147             users_collection.update_many({'_id': {'$ne':
                    new_user_id}}, {'$push': {'friends': {'username':
                    username, 'user_id': str(new_user_id)}}})
148
149             # Initialize empty entries in other collections
150             users_settings_collection.insert_one({'user_id':
                    new_user_id, 'settings': {}})
151             users_data_collection.insert_one({'user_id':
                    new_user_id, 'data': {}})
152             users_messages_collection.insert_one({'user_id':
                    new_user_id, 'messages': {}})
153
154             return redirect(url_for('login'))
155         else:
156             # Serve the static HTML file for the account creation
                    form
157             return app.send_static_file('CreateAccount.html')
158
159     def allowed_profile_pic(filename):
160         return '.' in filename and filename.rsplit('.', 1)[1].
                lower() in {'jpg', 'jpeg'}
161
162     ###############
163     #   LOGIN     #
164     ###############
165     @app.route('/login', methods=['GET', 'POST'])
166     def login():
167         if request.method == 'POST':
168             username = request.form['username']
169             password = request.form['password']
170             user_document = users_collection.find_one({'username':
                    username})
171
172             if user_document and check_password_hash(user_document
                    .get('password'), password):
173                 # Login successful
174                 session["logged_in"] = True
175                 session["username"] = username
176                 session["user_id"] = str(user_document['_id'])
177                 users_collection.update_one({'_id': user_document[
                        '_id']}, {'$set': {'last_login': datetime.now()
                        }})
178                 return redirect(url_for('plants'))
179             else:
180                 # Incorrect credentials, redirect with error
                        message
181                 return redirect(url_for('login', msg='Incorrect
                        credentials.'))
182         else:
183             # Serve the login page
184             return app.send_static_file('Luli.html')
```

```
185
186     ###############
187     #   LOGOUT    #
188     ###############
189     @app.route('/logout', methods=['GET'])
190     def logout():
191         session.clear()
192         return redirect(url_for('login'))
193
194
195     ########################
196     #   PLANTS + SENSORS   #
197     ########################
198     @app.route('/plants')
199     def plants():
200         user_id = session.get('user_id')
201         if not user_id:
202             return redirect(url_for('login'))
203
204         sensor_data_document = users_data_collection.find_one({'
                user_id': ObjectId(user_id)})
205
206         if sensor_data_document and 'sensor_data' in
                sensor_data_document:
207             sensor_data = sensor_data_document['sensor_data']
208             latest_sensor_data = list(sensor_data.values())[-1] if
                    sensor_data else {}
209         else:
210             # Default empty data if there's no sensor data
                    available
211             latest_sensor_data = {
212                 'light': 'N/A',
213                 'temp': 'N/A',
214                 'humidity': 'N/A',
215                 'ph': 'N/A',
216                 'tank': 'N/A'
217             }
218
219         return render_template('Plants.html', sensor_data=
                latest_sensor_data)
220
221     ##############
222     #   FORUM    #
223     ##############
224     @app.route('/forum')
225     def forum():
226         posts = list(users_messages_collection.find())
227         return render_template('Forum.html', posts=posts)
228
229
230     @app.route('/add_post', methods=['POST'])
```

```
231     def add_post():
232         if 'user_id' not in session:
233             return redirect(url_for('login'))  # Redirect to login
                    if not logged in
234 
235         user_document = users_collection.find_one({'_id': ObjectId
                (session['user_id'])})
236         if not user_document:
237             return "User not found", 404
238 
239         message = request.form.get('message')
240         image = request.files.get('image')
241 
242         post = {'user_id': session['user_id'],
243                 'name': user_document.get('first_name'),
244                 'message': message,
245                 'created': datetime.now()}
246 
247         if image and allowed_file(image.filename):
248             try:
249                 # Generate a unique filename using UUID
250                 unique_filename = str(uuid.uuid4())
251                 file_extension = os.path.splitext(secure_filename(
                        image.filename))[1]
252                 filename = unique_filename + file_extension
253                 # The relative path for use in HTML src attribute
254                 relative_image_path = os.path.join('USER_IMAGES',
                        filename)
255                 # The absolute path to save the image file
256                 absolute_image_path = os.path.join(app.config['
                        IMAGE_UPLOAD_FOLDER'], filename)
257                 # Save the image
258                 image.save(absolute_image_path)
259                 # Add image path to the post dictionary
260                 post['image_path'] = relative_image_path  # Store
                        the relative path, not absolute
261                 print(f"Image saved with unique filename {filename
                        }")
262             except Exception as e:
263                 print(f"Failed to save image: {e}")
264                 return "Failed to save image", 500
265 
266         # Insert the post (with the image path if there's an image
                ) into the database
267         users_messages_collection.insert_one(post)
268         return redirect(url_for('forum'))
269 
270 
271     def allowed_file(filename):
272         ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}
```

```
273         # Check if there is an extension in the filename and if
                the extension is allowed
274         return '.' in filename and filename.rsplit('.', 1)[1].
                lower() in ALLOWED_EXTENSIONS
275
276
277     ################
278     #   FRIENDS    #
279     ################
280     @app.route('/friends')
281     def friends():
282         if 'user_id' not in session:
283             return redirect(url_for('login'))  # Ensure the user
                    is logged in
284
285         user_id = session['user_id']
286         user_document = users_collection.find_one({'_id': ObjectId
                (user_id)})
287
288         if 'friends' in user_document:
289             # Extract the user_id of each friend and convert to
                    ObjectId
290             friend_ids = [ObjectId(friend['user_id']) for friend
                    in user_document['friends']]
291             friends = users_collection.find({'_id': {'$in':
                    friend_ids}})
292             friends_list = list(friends)  # Convert cursor to list
                     for passing to template
293             return render_template('Friends.html', friends=
                    friends_list)
294         else:
295             # Handle case where no friends are stored or the field
                     is missing
296             return render_template('Friends.html', friends=[])
297
298
299
300
301     ##################
302     #   SETTINGS     #
303     ##################
304     @app.route('/save_settings', methods=['POST'])
305     def save_settings():
306         if 'user_id' not in session:
307             return redirect(url_for('login'))
308
309         user_id = session['user_id']
310         water_interval = request.form.get('water_interval')
311         led_duration = request.form.get('led_duration')
312         pump_duration = request.form.get('pump_duration')
313         # ... other settings values
```

```
314
315         users_settings_collection.update_one(
316             {'user_id': ObjectId(user_id)},
317             {'$set': {
318                 'water_interval': water_interval,
319                 'led_duration': led_duration,
320                 'pump_duration': pump_duration
321                 # ... other settings
322             }},
323             upsert=True
324         )
325         return redirect(url_for('settings_page'))
326
327
328     @app.route('/settings')
329     def settings_page():
330         if 'user_id' not in session:
331             return redirect(url_for('login'))
332
333         user_id = session['user_id']
334         settings = users_settings_collection.find_one({'user_id':
               ObjectId(user_id)})
335         print("GOT SETTINGS: ", settings)
336         if settings is None:
337             # Set defaults if no settings are found
338             print("settings is None!")
339             settings = {
340                 'water_interval': '12',
341                 'led_duration': '12',
342                 'pump_duration': '5',
343                 # Include defaults for any other settings
344             }
345         else:
346             # If settings exist, remove MongoDB's _id before
                   passing to template
347             settings.pop('_id', None)
348
349         return render_template('settings.html', settings=settings)
350
351     @app.route('/download_settings/<friend_id>')
352     def download_settings(friend_id):
353         print("SAVING SETTINGS FROM FRIEND: ", friend_id)
354         settings = users_settings_collection.find_one({'user_id':
               ObjectId(friend_id)})
355         if settings:
356             del settings['_id']  # Remove the MongoDB ID before
                   sending
357             return jsonify(settings)
358         # Now get current user's document
359         user_id = session['user_id']
```

```
360         user_settings = users_settings_collection.find_one({'
                user_id': ObjectId(user_id)})
361         # Now overwrite user_settings with settings from friend
362         # But don't overwrite the user_id
363         user_settings.update(settings)
364         return jsonify({'error': 'Settings not found'}), 404
365
366
367     ##################
368     # CONTROL PANEL #
369     ##################
370     @app.route('/control_panel')
371     def control_panel():
372         user_id = session.get('user_id')
373         if not user_id:
374             return redirect(url_for('login'))
375         user_data = users_collection.find_one({'_id': ObjectId(
                user_id)})
376         device_id = user_data.get('device_id') if user_data else
                None
377         if device_id:
378             return render_template('ControlPanel.html', device_id=
                    device_id)
379         else:
380             return "Device ID not found", 404
381
382
383
384     if __name__ == "__main__":
385         ### How to run this file ###
386         """
387         1. Create a virtualenv and install packages
388             $ pip3 install virtualenv           [ only run this
                    once ]
389             $ virtualenv venv                   [ this create a
                    virtual python environment called "venv" ]
390             $ activate venv/bin/activate        [ mac only, this
                    is how to enable the venv so it doesnt install
                    packages to your global python ]
391             $ pip install -r requirements.txt   [ this installs
                    the necessary modules for the code to run (i.e.
                    Flask) ]
392         2. cd Luli/backend
393         3. Make sure MongoDB is running         [ On Mac: brew
                services start mongodb-community ]
394         4. python main.py
395         5. View the url in the output           [ If running
                locally: http://127.0.0.1:9696 ]
396         6. Open it in browser
397         """
398
```

```
399
400         print("Starting Flask server...")
401         app.run(debug=True, port=9696, host="0.0.0.0")
```

Listing 11: Main Flask Backend Code

A.2.2 api.py

```
1     from flask import request, jsonify
2     from bson.objectid import ObjectId
3     import datetime
4     from functools import wraps
5
6     ##############################
7     #        API FUNCTIONS       #
8     ##############################
9     '''
10        List of Valid API Endpoints and Their Methods:
11        ENDPOINT:                       METHODS:
12        - /api/test                     GET
13        - /api/get_hardware_id          GET
14        - /api/set_hardware_id          POST
15        - /api/update_settings          POST
16        - /api/get_settings             GET
17        - /api/get_humidity_data        GET
18        - /api/get_uv_data              GET
19        - /api/get_ph_data              GET
20        - /api/get_temp_data            GET
21        - /api/get_tank_data            GET
22        - /api/get_all_data             GET
23        - /api/update_uv_data           POST
24        - /api/update_ph_data           POST
25        - /api/update_temp_data         POST
26        - /api/update_humidity_data     POST
27        - /api/update_tank_data         POST
28        - /api/update_all_sensors_data  POST
29        - /api/update_manual_override   POST
30        - /api/get_manual_override      GET
31    '''
32
33    # Do it all in a function here because we can pass it "app"
          which is the same "app" as main.py
34    # this way avoids circular imports and lets us clean up the
          main.py file and keep the API defined here
35
36
37
38    def register_api_routes(app, users_settings_collection,
          users_data_collection, users_collection):
39        # This dictionary will store the manual override commands
              for each device by its ID
```

```
40          device_override_commands = {}
41
42
43          def ensure_object_id(obj_id):
44              if isinstance(obj_id, str):
45                  try:
46                      return ObjectId(obj_id)
47                  except Exception as e:
48                      print(f"Error converting string to ObjectId: {
                            e}")
49                      return None  # or handle the error as needed
50              elif isinstance(obj_id, ObjectId):
51                  return obj_id
52              else:
53                  print("Invalid type for user ID.")
54                  return None  # or handle as needed
55
56          def validate_API_key(device_id):
57              # Device ID will come from: request.headers.get('X-
                    Device-ID')
58              if device_id is None:
59                  return False, None, jsonify({'message': 'X-Device-
                        ID is required'}), 400
60              else:
61                  # Check if the device ID is a string
62                  if not isinstance(device_id, str):
63                      return False, None, jsonify({'message': '
                            Invalid X-Device-ID format'}), 400
64
65                  # Check if the length of the hardware ID is
                        correct
66                  if len(device_id) != 16:  # Assuming 16 characters
                        for a 128-bit ID
67                      return False, None, jsonify({'message': '
                            Invalid X-Device-ID format'}), 400
68
69                  # Check if the hardware ID contains only
                        hexadecimal characters
70                  valid_chars = set("0123456789abcdefABCDEF")
71                  if not all(char in valid_chars for char in
                        device_id):
72                      return False, None, jsonify({'message': '
                            Invalid X-Device-ID format'}), 400
73
74              user = users_data_collection.find_one({"device_id":
                    device_id})
75              if user is None:
76                  return False, None, jsonify({'message': '
                        Unauthorized - Device not recognized'}), 401
77
```

```
78              return True, ensure_object_id(user['_id']), jsonify({'
                    message': 'Authentication Successful'}), 200
79
80          # Custom Decorator to require an API key on certain routes
81          def require_api_key(f):
82              @wraps(f)
83              def decorated_function(*args, **kwargs):
84                  valid_api_key, user_id, message, status_code =
                        validate_API_key(request.headers.get('X-Device-
                        ID'))
85                  if not valid_api_key:
86                      return jsonify({'message': message}),
                            status_code
87                  return f(*args, **kwargs, user_id=user_id)
88              return decorated_function
89
90          # Custom test function to manually add data to the
                database
91          def add_sensor_data(client, database_name, collection_name
                , document_id, sensor_data):
92              # Access the MongoDB database
93              db = client[database_name]
94              collection = db[collection_name]
95
96              # Timestamp as a string in ISO format, with colons
                    replaced by underscores
97              #timestamp = datetime.utcnow().isoformat().replace
                    ('.', ':') + '+00_00'
98              timestamp = datetime.now().isoformat().replace('.', ':
                    ')
99
100             # Update the document with new sensor data at the
                    specified timestamp
101             print(f"TIMESTAMP IS: {timestamp}")
102             update_result = collection.update_one(
103                 {"_id": document_id},
104                 {"$set": {f"sensor_data.{timestamp}": sensor_data
                        }},
105                 upsert=True
106             )
107
108             # Check the result of the update
109             if update_result.upserted_id is not None:
110                 print(f"Document was created with ID: {
                        update_result.upserted_id}")
111             elif update_result.modified_count == 1:
112                 print("Update successful.")
113             else:
114                 print("Update failed or no changes made.")
115
116
```

```
117
118         ## USER SETTINGS
119         @app.route('/api/get_settings', methods=['GET'])
120         @require_api_key
121         def api_get_settings(user_id):
122             user_settings = users_settings_collection.find_one({'
                user_id': user_id})
123             if not user_settings:
124                 return "No settings found for this user."
125             return user_settings['settings']
126
127         @app.route('/api/update_settings', methods=['POST'])
128         @require_api_key
129         def api_update_settings(user_id):
130             user_settings = users_settings_collection.find_one({'
                user_id': user_id})
131             if not user_settings:
132                 users_settings_collection.insert_one({'user_id':
                    user_id, 'settings': request.json})
133             else:
134                 users_settings_collection.update_one({'user_id':
                    user_id}, {'$set': {'settings': request.json}})
135             return "Settings updated successfully."
136
137         @app.route('/api/set_hardware_id', methods=['POST'])
138         def api_set_hardware_id():
139             # Get the username and device_id passed in the request
                json:
140             payload = request.get_json()
141             username = payload.get('username')
142             device_id = payload.get('device_id')
143             user_credentials = users_collection.find_one({'
                username': username})
144             if not user_credentials:
145                 return jsonify({'message': 'User not found'}), 404
146             else:
147                 users_collection.update_one({'username': username
                    }, {'$set': {'device_id': device_id}})
148             return jsonify({'message': 'Hardware ID updated
                successfully'}), 200
149
150         @app.route('/api/get_hardware_id', methods=['GET'])
151         @require_api_key
152         def api_get_hardware_id(user_id):
153             user_credentials = users_collection.find_one({'user_id
                ': user_id})
154             if not user_credentials:
155                 return jsonify({'message': 'User not found'}), 404
156             return user_credentials['hardware_id']
157
158         ## SENSOR READINGS - GET
```

```
159
160         @app.route('/api/get_light_data', methods=['GET'])
161         @require_api_key
162         def api_get_light_data(user_id):
163             user_data = users_data_collection.find_one({'user_id':
                    user_id})
164             if not user_data:
165                 #return "No data found for this user."
166                 return jsonify({'message': 'No data found for this
                        user'}), 404
167             return user_data['light']
168
169         @app.route('/api/get_ph_data', methods=['GET'])
170         @require_api_key
171         def api_get_ph_data(user_id):
172             user_data = users_data_collection.find_one({'user_id':
                    user_id})
173             if not user_data:
174                 #return "No data found for this user."
175                 return jsonify({'message': 'No data found for this
                        user'}), 404
176             return user_data['ph']
177
178         @app.route('/api/get_temp_data', methods=['GET'])
179         @require_api_key
180         def api_get_temp_data(user_id):
181             user_data = users_data_collection.find_one({'user_id':
                    user_id})
182             if not user_data:
183                 #return "No data found for this user."
184                 return jsonify({'message': 'No data found for this
                        user'}), 404
185             return user_data['temp']
186
187         @app.route('/api/get_humidity_data', methods=['GET'])
188         @require_api_key
189         def api_get_humidity_data(user_id):
190             user_data = users_data_collection.find_one({'user_id':
                    user_id})
191             if not user_data:
192                 #return "No data found for this user."
193                 return jsonify({'message': 'No data found for this
                        user'}), 404
194             return user_data['humidity']
195
196         @require_api_key
197         def api_get_tank_data(user_id):
198             user_data = users_data_collection.find_one({'user_id':
                    user_id})
199             if not user_data:
200                 #return "No data found for this user."
```

```
201                 return jsonify({'message': 'No data found for this
                        user'}), 404
202             return user_data['tank']
203
204         @app.route('/api/get_all_data', methods=['GET'])
205         @require_api_key
206         def api_get_all_data(user_id):
207             user_data = users_data_collection.find_one({'user_id':
                    user_id})
208             if not user_data:
209                 #return "No data found for this user."
210                 return jsonify({'message': 'No data found for this
                        user'}), 404
211             return user_data
212
213         ## SENSOR READINGS - POST
214         @app.route('/api/update_light_data', methods=['POST'])
215         @require_api_key
216         def api_update_light_data(user_id):
217             user_data = users_data_collection.find_one({'user_id':
                    user_id})
218             if not user_data:
219                 users_data_collection.insert_one({'user_id':
                        user_id, 'light': request.json})
220             else:
221                 users_data_collection.update_one({'user_id':
                        user_id}, {'$set': {'light': request.json}})
222             #return "Light Data updated successfully."
223             return jsonify({'message': 'Light Data updated
                    successfully'}), 200
224
225         @app.route('/api/update_ph_data', methods=['POST'])
226         @require_api_key
227         def api_update_pH_data(user_id):
228             user_data = users_data_collection.find_one({'user_id':
                    user_id})
229             if not user_data:
230                 users_data_collection.insert_one({'user_id':
                        user_id, 'ph': request.json})
231             else:
232                 users_data_collection.update_one({'user_id':
                        user_id}, {'$set': {'ph': request.json}})
233             #return "pH Data updated successfully."
234             return jsonify({'message': 'pH Data updated
                    successfully'}), 200
235
236         @app.route('/api/update_temp_data', methods=['POST'])
237         @require_api_key
238         def api_update_temp_data(user_id):
239             user_data = users_data_collection.find_one({'user_id':
                    user_id})
```

```
240             if not user_data:
241                 users_data_collection.insert_one({'user_id':
                        user_id, 'temp': request.json})
242             else:
243                 users_data_collection.update_one({'user_id':
                        user_id}, {'$set': {'temp': request.json}})
244             #return "Temperature Data updated successfully."
245             return jsonify({'message': 'Temperature Data updated
                    successfully'}), 200
246
247         @app.route('/api/update_humidity_data', methods=['POST'])
248         @require_api_key
249         def api_update_humidity_data(user_id):
250             user_data = users_data_collection.find_one({'user_id':
                    user_id})
251             if not user_data:
252                 users_data_collection.insert_one({'user_id':
                        user_id, 'humidity': request.json})
253             else:
254                 users_data_collection.update_one({'user_id':
                        user_id}, {'$set': {'humidity_data': request.
                        json}})
255             #return "Humidity Data updated successfully."
256             return jsonify({'message': 'Humidity Data updated
                    successfully'}), 200
257
258         @app.route('/api/update_tank_data', methods=['POST'])
259         @require_api_key
260         def api_update_tank_data(user_id):
261             user_data = users_data_collection.find_one({'user_id':
                    user_id})
262             if not user_data:
263                 users_data_collection.insert_one({'user_id':
                        user_id, 'tank': request.json})
264             else:
265                 users_data_collection.update_one({'user_id':
                        user_id}, {'$set': {'tank': request.json}})
266             #return "Tank Data updated successfully."
267             return jsonify({'message': 'Tank Data updated
                    successfully'}), 200
268
269
270         @app.route('/api/update_all_sensor_data', methods=['POST'
                ])
271         @require_api_key
272         def update_all_sensor_data(user_id):
273             sensor_data = request.get_json()
274             if not sensor_data:
275                 return jsonify({'message': 'No sensor data
                        provided'}), 400
```

```
276             # Update the document with new sensor data at the
                    specified timestamp
277             print(f"TIMESTAMP IS: {timestamp}")
278             timestamp = datetime.datetime.now().isoformat().
                    replace('.', ':')
279             # Check if the user already has sensor data
280             if users_data_collection.count_documents({'_id':
                    user_id, 'sensor_data': {'$exists': True}}) == 0:
281                 # If not, create a new document with sensor_data
                        as a dictionary
282                 users_data_collection.insert_one({'_id': user_id,
                        'sensor_data': {timestamp: sensor_data}})
283             else:
284                 # If sensor_data exists, update the existing
                        document with new sensor data at the specified
                        timestamp
285                 users_data_collection.update_one(
286                     {"_id": user_id},
287                     {"$set": {f"sensor_data.{timestamp}":
                            sensor_data}},
288                     upsert=True
289                 )
290
291             #return "All sensor data updated successfully."
292             return jsonify({'message': 'All sensor data updated
                    successfully'}), 200
293
294         @app.route('/api/test', methods=['GET'])
295         def api_test():
296             return "API is working."
297
298         @app.route('/api/update_manual_override/<device_id>',
                methods=['POST'])
299         def update_manual_override(device_id):
300             content = request.json
301             if device_id not in device_override_commands:
302                 device_override_commands[device_id] = {}
303             device_override_commands[device_id].update(content)
304             return jsonify(success=True, commands=
                    device_override_commands[device_id]), 200
305
306         @app.route('/api/get_manual_override/<device_id>', methods
                =['GET'])
307         def get_manual_override(device_id):
308             commands = device_override_commands.get(device_id, {})
309             # Clear the commands after they have been fetched
310             device_override_commands[device_id] = {}
311             return jsonify(commands), 200
```

Listing 12: Flask Backend API Code

A.2.3 messaging.py

```
from pymongo import MongoClient
from pymongo.errors import DuplicateKeyError
from bson.objectid import ObjectId

def follow_user(FROM_user_id, TO_user_id, mongoclient_url='mongodb://localhost:27017'):
    user_id = ObjectId(FROM_user_id)
    user_to_follow_id = ObjectId(TO_user_id)

    client = MongoClient(mongoclient_url)
    DATABASE = client['user_data']
    users_collection = DATABASE.credentials

    # Make sure users exist
    user = users_collection.find_one({'_id': user_id})
    user_to_follow = users_collection.find_one({'_id': user_to_follow_id})

    # If both users exist and the user_to_follow is not already in the followed_users list, add the user_to_follow to the user's friends list
    print(user_to_follow_id not in user.get('followed_users', []))
    if user and user_to_follow and user_to_follow_id not in user.get('followed_users', []):
        # Check if the user already has a 'followed_users' field
        if 'followed_users' in user:
            # If the 'followed_users' field exists, append the user_to_follow_id to the list
            users_collection.update_one({'_id': user_id}, {'$push': {'followed_users': user_to_follow_id}})
        else:
            # If the 'followed_users' field does not exist, create a new list with the user_to_follow_id
            users_collection.update_one({'_id': user_id}, {'$set': {'followed_users': [user_to_follow_id]}})
        return True
    elif user_to_follow_id in user.get('followed_users', []):
        return True
    return False

def get_followed_users(user_id, mongoclient_url='mongodb://localhost:27017'):
    user_id = ObjectId(user_id)

    client = MongoClient(mongoclient_url)
    DATABASE = client['user_data']
```

```
37          users_collection = DATABASE.credentials
38
39          user = users_collection.find_one({'_id': user_id})
40          if user:
41              return user.get('followed_users', [])
42          return None
43
44      def message_user(FROM_user_id, TO_user_id, message,
            mongoclient_url='mongodb://localhost:27017'):
45          user_id = ObjectId(FROM_user_id)
46          user_to_message_id = ObjectId(TO_user_id)
47
48          client = MongoClient(mongoclient_url)
49          DATABASE = client['user_data']
50          users_collection = DATABASE.credentials
51          users_messages_collection = DATABASE.messages
52
53          # Make sure users exist
54          user = users_collection.find_one({'_id': user_id})
55          user_to_message = users_collection.find_one({'_id':
                user_to_message_id})
56
57          # If both users exist, add the message to the database
58          if user and user_to_message:
59              users_messages_collection.insert_one({
60                  'sender_id': user_id,
61                  'receiver_id': user_to_message_id,
62                  'message': message
63              })
64              return True
65          return False
66
67      def get_most_recent_message(FROM_user_id, TO_user_id,
            mongoclient_url='mongodb://localhost:27017'):
68          user_id = ObjectId(FROM_user_id)
69          user_to_message_id = ObjectId(TO_user_id)
70
71          client = MongoClient(mongoclient_url)
72          DATABASE = client['user_data']
73          users_messages_collection = DATABASE.messages
74
75          # Query by 'receiver_id' instead of 'user_id'
76          user_messages = users_messages_collection.find({'
                receiver_id': user_to_message_id})
77
78          user_messages_list = list(user_messages)
79          if len(user_messages_list) == 0:
80              return None
81          return user_messages_list[-1]
82
```

```
83      def get_most_recent_message_from_user(FROM_user_id, TO_user_id
            , mongoclient_url='mongodb://localhost:27017'):
84          user_id = ObjectId(FROM_user_id)
85          user_to_message_id = ObjectId(TO_user_id)
86
87          client = MongoClient(mongoclient_url)
88          DATABASE = client['user_data']
89          users_messages_collection = DATABASE.messages
90
91          # Query by 'sender_id' instead of 'user_id'
92          user_messages = users_messages_collection.find({'sender_id
                ': user_id, 'receiver_id': user_to_message_id})
93
94          user_messages_list = list(user_messages)
95          if len(user_messages_list) == 0:
96              return None
97          return user_messages_list[-1]
98
99      def user_id_to_username(user_id, mongoclient_url='mongodb://
            localhost:27017'):
100         user_id = ObjectId(user_id)
101
102         client = MongoClient(mongoclient_url)
103         DATABASE = client['user_data']
104         users_collection = DATABASE.credentials
105
106         user = users_collection.find_one({'_id': user_id})
107         if user:
108             return user['username']
109         return None
110
111     def username_to_user_id(username, mongoclient_url='mongodb://
            localhost:27017'):
112         client = MongoClient(mongoclient_url)
113         DATABASE = client['user_data']
114         users_collection = DATABASE.credentials
115
116         userid = users_collection.find_one({'username': username})
117         if userid:
118             return userid['_id']
119
120
121     def test_follow_user():
122         user_id = username_to_user_id('Liam_TEST')
123         user_to_follow_id = username_to_user_id('Luigi_TEST')
124         assert follow_user(user_id, user_to_follow_id) == True
125         assert follow_user(user_to_follow_id, user_id) == True
126
127
128     def test_check_messages():
```

```
129         user_id = ObjectId('6615f9b445f79acfdde258eb')  # Liam_TEST
130         user_to_message_id = ObjectId('6615f98545f79acfdde258ea')  # Luigi_TEST
131         message = "Hello, how are you?!!!"
132         message_user(user_id, user_to_message_id, message)
133         client = MongoClient()
134         DATABASE = client['user_data']
135 
136         users_messages_collection = DATABASE.messages
137         # Query by 'receiver_id' instead of 'user_id'
138         user_messages = users_messages_collection.find({'receiver_id': user_to_message_id})
139 
140         user_messages_list = list(user_messages)
141         assert len(user_messages_list) != 0
142         assert message in [user_message['message'] for user_message in user_messages_list]
143 
144 
145     if __name__ == "__main__":
146         test_follow_user()
147         test_check_messages()
148 
149 
150         assert user_id_to_username(username_to_user_id('Liam_TEST')) == 'Liam_TEST'
151         client = MongoClient()
152         DATABASE = client['user_data']
153 
154         messages = DATABASE.messages.find({'receiver_id': ObjectId(username_to_user_id('Luigi_TEST'))})
155         message_values = [message['message'] for message in messages]
156         print("Messages for Luigi_TEST to read: ", message_values)
157 
158         messages = DATABASE.messages.find({'receiver_id': ObjectId(username_to_user_id('Liam_TEST'))})
159         message_values = [message['message'] for message in messages]
160         print("Messages for Liam_TEST to read: ", message_values)
161 
162         print("\n")
163         most_recent_message = get_most_recent_message(username_to_user_id('Liam_TEST'), username_to_user_id('Luigi_TEST'))
164         if most_recent_message:
165             sender_username = user_id_to_username(most_recent_message['sender_id'])
166             print(f"Most recent message for Luigi_TEST from {sender_username}: {most_recent_message['message']}"
```

```
            )
167         else:
168             print("No recent message for Luigi_TEST")
169         print("\n")
170 
171         most_recent_message = get_most_recent_message_from_user(
               username_to_user_id('Liam_TEST'), username_to_user_id('
               Luigi_TEST'))
172         if most_recent_message:
173             print(f"Most recent message from Liam_TEST to
                   Luigi_TEST: {most_recent_message['message']}")
174         else:
175             print("No recent message from Liam_TEST to Luigi_TEST"
                   )
176 
177         followed_users = get_followed_users(username_to_user_id('
               Liam_TEST'))
178         usernames = [user_id_to_username(user_id) for user_id in
               followed_users]
179         print(f"Liam_TEST follows: {usernames}")
180 
181         print("All tests passed!")
```

Listing 13: Messaging Functions for MongoDB

A.2.4 test_save_sensor_data.py

```
1     from pymongo import MongoClient
2     from datetime import datetime
3     from bson import ObjectId
4 
5     def add_sensor_data(client, database_name, collection_name,
         document_id, sensor_data):
6         # Access the MongoDB database
7         db = client[database_name]
8         collection = db[collection_name]
9 
10        # Timestamp as a string in ISO format, with colons
             replaced by underscores
11        #timestamp = datetime.utcnow().isoformat().replace('.',
             ':') + '+00_00'
12        timestamp = datetime.now().isoformat().replace('.', ':')
13 
14        # Update the document with new sensor data at the
             specified timestamp
15        print(f"TIMESTAMP IS: {timestamp}")
16        update_result = collection.update_one(
17            {"_id": document_id},
18            {"$set": {f"sensor_data.{timestamp}": sensor_data}},
19            upsert=True
20        )
```

```
21
22          # Check the result of the update
23          if update_result.upserted_id is not None:
24              print(f"Document was created with ID: {update_result.
                    upserted_id}")
25          elif update_result.modified_count == 1:
26              print("Update successful.")
27          else:
28              print("Update failed or no changes made.")
29
30      # Example usage
31      client = MongoClient("mongodb://localhost:27017/")
32      database_name = 'user_data'
33      collection_name = 'data'
34      document_id = ObjectId('65fa4c38972991d86e690014')
35      sensor_data = {
36          "temp": 25.45,
37          "humidity": 63,
38          "light": 25,
39          "tank": 36,
40          "ph": 7
41      }
42
43      add_sensor_data(client, database_name, collection_name,
            document_id, sensor_data)
```

Listing 14: Debugging Code for Saving Sensor Data

A.3 Frontend HTML/CSS/Javascript Code

```
1
2       <!-- control_panel.html -->
3       <!DOCTYPE html>
4       <html lang="en">
5       <head>
6       <meta charset="UTF-8">
7       <title>Control Panel - LuLi Hydroponics</title>
8       <link href="{{ url_for('static', filename='style.css') }}" rel
           ="stylesheet">
9       <script src="{{ url_for('static', filename='scripts/
           controlPanel.js') }}"></script>
10      </head>
11      <body>
12      <header>
13          <img class="logo" src = "{{ url_for('static', filename='
               images/hydroponic.png')}}" alt = "hydroPlant"  width =
               50 height =50 style = "float: left;" >
14          <img src = "{{ url_for('static', filename='images/
               luli_logo.png')}}" alt="Luli Hydroponics" width = 60
               height= 60 style="float: left">
15          <div class = "corner_topRight"> </div>
16      </header>
17      <nav class = "nav_list">
18
19          <div class = "button" >
20              <a href="/" target=" _self"><img src = "{{ url_for('
                   static', filename='images/user.png')}}" alt = "login
                   " height= 50 width = 50></a>
21              <label class="button_label"> Login </label>
22          </div>
23          <div class = "button" >
24              <a href="/plants" target=" _self"> <img src = "{{
                   url_for('static', filename='images/pot.png')}}" alt
                   = "plant" height= 50 width = 50> </a>
25              <label class="button_label"> Plants </label>
26          </div>
27          <div class = "button" >
28              <a href="/friends" target=" _self"> <img src = "{{
                   url_for('static', filename='images/friends.png')}}"
                   alt = "plant" height= 50 width=50> </a>
29              <label class="button_label"> Friends </label>
30          </div>
31          <div class = "button" >
32              <a href="/forum" target=" _self"> <img src = "{{
                   url_for('static', filename='images/discussion.png')
                   }}" alt = "plant" height= 50 width=50> </a>
33              <label class="button_label"> Forum </label>
34          </div>
35          <div class = "button" >
```

```
36            <a href="/settings" target=" _self"> <img src = "{{
                 url_for('static', filename='images/setting.png')}}"
                 alt = "plant" height= 50 width=50> </a>
37            <label class="button_label"> Settings </label>
38        </div>
39            <div class = "corner_bottomLeft"> </div>
40       </nav>
41    <article>
42        <div id="controls">
43            <button onclick="sendCommand('motor', 'on')">Turn
                 Motor On</button>
44            <button onclick="sendCommand('motor', 'off')">Turn
                 Motor Off</button>
45            <button onclick="sendCommand('leds', 'on')">Turn LEDs
                 On</button>
46            <button onclick="sendCommand('leds', 'off')">Turn LEDs
                  Off</button>
47        </div>
48    </article>
49    <script>
50        function sendCommand(device, action) {
51            const data = { [device]: action };
52            fetch(`/api/update_manual_override/{{ device_id }}`, {
53                method: 'POST',
54                headers: { 'Content-Type': 'application/json' },
55                body: JSON.stringify(data)
56            })
57            .then(response => response.json())
58            .then(data => console.log('Command sent:', data))
59            .catch(error => console.error('Error sending command
                 :', error));
60        }
61    </script>
62    </body>
63    </html>
```

Listing 15: Control Panel Html

```
<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title> LuLi Hydroponics </title>

<link href="style.css"
      rel = "stylesheet">

</head>

<body>

   <header>
      <img class="logo" src = "images/hydroponic.png" alt = "
         hydroPlant"  width = 50 height =50 style = "float:
         left;" >
      <img src = "images/luli_logo.png" alt="Luli Hydroponics"
          width = 60 height= 60 style="float: left">
      <div class = "corner_topRight"> </div>
   </header>

   <nav class = "nav_list">

    <div class = "button" >
       <a href="/" target=" _self"><img src = "images/user.png
          " alt = "login" height= 50 width = 50></a>
       <label class="button_label"> Login </label>
    </div>
    <div class = "button" >
       <a href="/plants" target=" _self"> <img src = "images/
          pot.png" alt = "plant" height= 50 width = 50> </a>
       <label class="button_label"> Plants </label>
    </div>
    <div class = "button" >
       <a href="/friends" target=" _self"> <img src = "images/
          friends.png" alt = "plant" height= 50 width=50> </a>
       <label class="button_label"> Friends </label>
    </div>
    <div class = "button" >
       <a href="/forum" target=" _self"> <img src = "images/
          discussion.png" alt = "plant" height= 50 width=50> <
          /a>
       <label class="button_label"> Forum </label>
    </div>
    <div class = "button" >
       <a href="/settings" target=" _self"> <img src = "images
          /setting.png" alt = "plant" height= 50 width=50> </a
          >
       <label class="button_label"> Settings </label>
```

```
41          </div>
42             <div class = "corner_bottomLeft"> </div>
43         </nav>
44
45     <article>
46
47           <div id = "createAccount" class="createAccount">
48             <div class = "circleLogo">
49                 <div class = "logoFrontPage">
50                     <img src = "images/luli_logo.png" alt="Luli
                           Hydroponics" width = 160 height= 160 style=
                           "float: center" >
51                 </div>
52                 <div></div>
53
54             </div>
55
56             <form id="createAccountForm" class = "
                  createAccountForm" action = "/create_account"
                  method = "POST" enctype="multipart/form-data">
57
58                 <div id = "welcomeMessage" class = "welcomeMessage
                      ">Welcome to Luli Hydroponics!</div>
59
60                 <div class="formGroup">
61                     <label for="first-name">First Name:</label>
62                     <input type="text" id="first-name" name="first-
                          name" required>
63                 </div>
64                 <div class="formGroup">
65                     <label for="last-name">Last Name:</label>
66                     <input type="text" id="last-name" name="last-
                          name" required>
67                 </div>
68                 <div class="formGroup">
69                     <label for="username">Username:</label>
70                     <input type="text" id="username" name="username
                          " required>
71                 </div>
72                 <div class="formGroup">
73                     <label for="password">Password:</label>
74                     <input type="password" id="password" name="
                          password" required>
75                 </div>
76                 <div class="formGroup">
77                     <label for="profile_pic">Profile Picture:</
                          label>
78                     <input type="file" id="profile_pic" name="
                          profile_pic" accept="image/jpeg, image/jpg">
79                 </div>
80                 <button type="submit" class = "createAccountButton
```

```
                    ">Create Account</button>
81              </form>
82
83            </div>
84      </article>
85
86      </body>
87      </html>
```

Listing 16: Create Account HTML

```
<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title> LuLi Hydroponics </title>

<link href="style.css"
      rel = "stylesheet">


</head>

<body>

   <header>
      <img class="logo" src = "images/hydroponic.png" alt = "
         hydroPlant"  width = 50 height =50 style = "float:
         left;" >
      <img src = "images/luli_logo.png" alt="Luli Hydroponics"
          width = 60 height= 60 style="float: left">
      <div class = "corner_topRight"> </div>
   </header>

   <nav class = "nav_list">

      <div class = "button" >
         <a href="/" target=" _self"><img src = "images/user.
            png" alt = "login" height= 50 width = 50></a>
         <label class="button_label"> Login </label>
      </div>
      <div class = "button" >
         <a href="/plants" target=" _self"> <img src = "images
            /pot.png" alt = "plant" height= 50 width = 50> </a
            >
         <label class="button_label"> Plants </label>
      </div>
      <div class = "button" >
         <a href="/friends" target=" _self"> <img src = "
            images/friends.png" alt = "plant" height= 50 width
            =50> </a>
         <label class="button_label"> Friends </label>
      </div>
      <div class = "button" >
         <a href="/forum" target=" _self"> <img src = "images/
            discussion.png" alt = "plant" height= 50 width=50>
             </a>
         <label class="button_label"> Forum </label>
      </div>
      <div class = "button" >
         <a href="/settings" target=" _self"> <img src = "
```

```
            images/setting.png" alt = "plant" height= 50 width
            =50> </a>
41          <label class="button_label"> Settings </label>
42       </div>
43          <div class = "corner_bottomLeft"> </div>
44       </nav>
45
46       <article>
47        <div id="forumContainer" class="forumContainer">
48          <div id="forumPosts" class="forumPosts">
49             {% for post in posts %}
50                <div class="post">
51                   <div class="name" style="margin:15px">{{
                        post.name }}</div>
52                   <!-- Check if 'created' exists before using
                        it -->
53                   {% if post.get('created') %}
54                      <div class="date" style="margin:15px">{{
                           post.created.strftime('%m/%d/%y %I:%M%
                           p') }}</div>
55                   {% else %}
56                      <div class="date" style="margin:15px">
                           Date not available</div>
57                   {% endif %}
58                   <div class="message" style="margin:15px">{{
                        post.message }}</div>
59                   {% if post.image_path %}
60                      <img src="{{ url_for('static', filename=
                           post.image_path) }}" alt="Post Image"
                           style="max-width: 500px;">
61                   {% endif %}
62                </div>
63                {% endfor %}
64          </div>
65          <div id="forumMenu" class="forumMenu">
66             <button id="addPostButton">Add Post</button>
67          </div>
68        </div>
69
70        <!-- Modal for Adding a New Post -->
71        <div id="postModal" class="modal">
72          <div class="modal-content">
73              <span class="close">&times;</span>
74              <form action="/add_post" method="post" enctype="
                   multipart/form-data">
75                 <label for="message">Message:</label>
76                 <textarea id="message" name="message"
                      required></textarea><br>
77                 <label for="image">Image (optional):</label>
78                 <input type="file" id="image" name="image"><
                      br>
```

```
79                        <button type="submit">Submit</button>
80                    </form>
81                </div>
82            </div>
83        </article>
84
85        <script src="scripts/modal.js"></script>
86
87    </body>
88
89
90
91    </html>
```

Listing 17: Forum HTML

```
1
2     <!DOCTYPE html>
3     <html lang = "en">
4     <head>
5     <meta charset = "UTF-8">
6     <title> LuLi Hydroponics </title>
7
8     <link href="{{ url_for('static', filename='style.css') }}" rel
         ="stylesheet">
9
10    </head>
11
12    <body>
13
14       <header>
15          <img class="logo" src = "{{ url_for('static', filename='
               images/hydroponic.png')}}" alt = "hydroPlant"  width
               = 50 height =50 style = "float: left;" >
16          <img src = "{{ url_for('static', filename='images/
               luli_logo.png')}}" alt="Luli Hydroponics" width = 60
               height= 60 style="float: left">
17          <div class = "corner_topRight"> </div>
18       </header>
19
20       <nav class = "nav_list">
21
22          <div class = "button" >
23             <a href="/" target=" _self"><img src = "{{ url_for('
                  static', filename='images/user.png')}}" alt = "
                  login" height= 50 width = 50></a>
24             <label class="button_label"> Login </label>
25          </div>
26          <div class = "button" >
27             <a href="/plants" target=" _self"> <img src = "{{
                  url_for('static', filename='images/pot.png')}}"
                  alt = "plant" height= 50 width = 50> </a>
28             <label class="button_label"> Plants </label>
29          </div>
30          <div class = "button" >
31             <a href="/friends" target=" _self"> <img src = "{{
                  url_for('static', filename='images/friends.png')}}
                  " alt = "plant" height= 50 width=50> </a>
32             <label class="button_label"> Friends </label>
33          </div>
34          <div class = "button" >
35             <a href="/forum" target=" _self"> <img src = "{{
                  url_for('static', filename='images/discussion.png
                  ')}}" alt = "plant" height= 50 width=50> </a>
36             <label class="button_label"> Forum </label>
37          </div>
```

```
38            <div class = "button" >
39               <a href="/settings" target=" _self"> <img src = "{{
                    url_for('static', filename='images/setting.png')}}
                    " alt = "plant" height= 50 width=50> </a>
40               <label class="button_label"> Settings </label>
41            </div>
42               <div class = "corner_bottomLeft"> </div>
43         </nav>
44
45         <article id="friendList" class="friendList">
46          {% for friend in friends %}
47          <div class="friendModule">
48              <div class="userProfile">
49                <img src="{{ url_for('static', filename='
                     PROFILE_PICS/' + friend.username + '.jpg') }}"
                     alt="profile pic" class="profileImage"/>
50                <div class="userName">{{ friend.username }}</div
                     >
51              </div>
52              <div class="userInfo">
53                  <p class="userAbout">Download my settings to
                       your Luli Hydroponics system!</p>
54                  <div class="userInteraction">
55                      <button type="button" class="
                           downloadSettings" onclick="
                           handleDownloadSettings(this)" data-friend
                           -id="{{ friend.user_id }}">
56                       <img src="{{ url_for('static', filename
                             ='images/download.png')}}" alt="
                             download" width="80%"/>
57                      </button>
58                  </div>
59              </div>
60          </div>
61          {% endfor %}
62       </article>
63
64        <script src="scripts/settingsDownload.js"></script>
65
66     </body>
67     </html>
```

Listing 18: Friends Page HTML

```
1
2     <!DOCTYPE html>
3     <html lang = "en">
4     <head>
5     <meta charset = "UTF-8">
6     <title> LuLi Hydroponics </title>
7
8     <link href="style.css"
9           rel = "stylesheet">
10
11
12    </head>
13    <script>
14       window.onload = function() {
15           const params = new URLSearchParams(window.location.
                search);
16           const msg = params.get('msg');
17           if (msg) {
18               alert(msg);
19           }
20       };
21    </script>
22
23    <body>
24
25       <header>
26          <img class="logo" src = "images/hydroponic.png" alt = "
               hydroPlant"  width = 50 height =50 style = "float:
               left;" >
27          <img src = "images/luli_logo.png" alt="Luli Hydroponics"
                width = 60 height= 60 style="float: left">
28          <div class = "corner_topRight"> </div>
29       </header>
30
31         <nav class = "nav_list">
32
33          <div class = "button" >
34             <a href="/" target=" _self"><img src = "images/user.
                  png" alt = "login" height= 50 width = 50></a>
35             <label class="button_label"> Login </label>
36          </div>
37          <div class = "button" >
38             <a href="/plants" target=" _self"> <img src = "images
                  /pot.png" alt = "plant" height= 50 width = 50> </a
                  >
39             <label class="button_label"> Plants </label>
40          </div>
41          <div class = "button" >
42             <a href="/friends" target=" _self"> <img src = "
                  images/friends.png" alt = "plant" height= 50 width
```

```
                =50> </a>
43              <label class="button_label"> Friends </label>
44           </div>
45           <div class = "button" >
46              <a href="/forum" target=" _self"> <img src = "images/
                   discussion.png" alt = "plant" height= 50 width=50>
                    </a>
47              <label class="button_label"> Forum </label>
48           </div>
49           <div class = "button" >
50              <a href="/settings" target=" _self"> <img src = "
                   images/setting.png" alt = "plant" height= 50 width
                   =50> </a>
51              <label class="button_label"> Settings </label>
52           </div>
53              <div class = "corner_bottomLeft"> </div>
54         </nav>
55
56         <article>
57          <form action="/login" method="POST">
58          <div class="module_login">
59                <div class = "user">
60                </div>
61                <div>
62                <label for="username" class="profile" > Username:<
                     /label>
63                <input type="text" id="username" name="username"
                     style="margin-bottom: 15px;">
64                </div>
65                <div>
66                <label for="password" class="password" >Password:<
                     /label>
67                <input type="password" id="password" name="
                     password" style="margin-bottom: 15px;">
68                </div>
69                <button type="submit">Login</button>
70                <a href = "/create_account"> <button type ="button
                     ">Create Account</button> </a>
71              </form>
72           </div>
73       </article>
74
75
76
77     </body>
78
79
80
81     </html>
```

Listing 19: Login Page HTML

```
<!DOCTYPE html>
<html lang = "en">
<head>
<meta charset = "UTF-8">
<title> LuLi Hydroponics </title>

<link href="{{ url_for('static', filename='style.css')}}"
      rel = "stylesheet">


</head>

<body>

   <header>
      <img class="logo" src = "{{ url_for('static', filename='
         images/hydroponic.png')}}" alt = "hydroPlant"  width
         = 50 height =50 style = "float: left;" >
      <img src = "{{ url_for('static', filename='images/
         luli_logo.png')}}" alt="Luli Hydroponics" width = 60
         height= 60 style="float: left">
      <div class = "corner_topRight"> </div>
   </header>

   <nav class = "nav_list">

      <div class = "button" >
         <a href="/" target=" _self"><img src = "{{ url_for('
            static', filename='images/user.png')}}" alt = "
            login" height= 50 width = 50></a>
         <label class="button_label"> Login </label>
      </div>
      <div class = "button" >
         <a href="/plants"><img src="{{ url_for('static',
            filename='images/pot.png')}}" alt="plant" height="
            50" width="50"></a>
         <label class="button_label"> Plants </label>
      </div>
      <div class = "button" >
         <a href="/friends" target=" _self"> <img src = "{{
            url_for('static', filename='images/friends.png')}}
            " alt = "plant" height= 50 width=50> </a>
         <label class="button_label"> Friends </label>
      </div>
      <div class = "button" >
         <a href="/forum" target=" _self"> <img src = "{{
            url_for('static', filename='images/discussion.png
            ')}}" alt = "plant" height= 50 width=50> </a>
         <label class="button_label"> Forum </label>
      </div>
```

```
39              <div class = "button" >
40                 <a href="/settings" target=" _self"> <img src = "{{
                      url_for('static', filename='images/setting.png')}}
                      " alt = "plant" height= 50 width=50> </a>
41                 <label class="button_label"> Settings </label>
42              </div>
43                 <div class = "corner_bottomLeft"> </div>
44            </nav>
45
46            <article>
47
48            <div class = "measurements">
49
50            <div class = "measurementBar">
51             <div class = "barProgress" style="
52             height: 20%;
53             width: 35px;
54             background: gold; ">
55             </div>
56             </div>
57
58            <div class = "measurementBar">
59             <div class = "barProgress" style="
60             height: 20%;
61             width: 35px;
62             background: red; ">
63             </div>
64             </div>
65
66            <div class = "measurementBar">
67             <div class = "barProgress" style="
68             height: 20%;
69             width: 35px;
70             background: aqua; ">
71             </div>
72             </div>
73
74            <div class = "measurementBar">
75             <div class = "barProgress" style="
76             height: 20%;
77             width: 35px;
78             background: orange; ">
79             </div>
80             </div>
81
82            <div class = "measurementBar">
83             <div class = "barProgress" style="
84             height: 20%;
85             width: 35px;
86             background:greenyellow; ">
87             </div>
```

```
88            </div>
89
90
91            <div class = "type"> {{sensor_data.light}} </div>
92            <div class = "type"> {{sensor_data.temp}}</div>
93            <div class = "type"> {{sensor_data.humidity}} </div>
94            <div class = "type"> {{sensor_data.ph}} </div>
95            <div class = "type"> {{sensor_data.tank}} </div>
96
97            <div class = "type"> LIGHT </div>
98            <div class = "type"> TEMP </div>
99            <div class = "type"> HUMIDITY </div>
100           <div class = "type"> pH </div>
101           <div class = "type"> TANK </div>
102
103           </div>
104
105          <div class = "plantSection">
106
107           <div class = "plantInfo">
108
109            <div>
110               <img src = "{{ url_for('static', filename='images/
                     plants/spinach.jpeg')}}" alt = "spinach" height =
                     90% width = 90% class = "plantImages">
111            </div>
112
113            <div> SPINACH</div>
114
115            <div>Container: </div>
116            <div> A </div>
117
118            <div>Planted: </div>
119            <div>5/25/24 </div>
120
121            <div>Harvest: </div>
122            <div>6/7/24 </div>
123
124           </div>
125
126           <div class = "plantInfo">
127
128            <div>
129               <img src = "{{ url_for('static', filename='images/
                     plants/kale.jpeg')}}" alt = "Kale" height = 90%
                     width = 90% class = "plantImages">
130            </div>
131
132            <div> Kale</div>
133
134            <div>Container: </div>
```

```
135             <div> A </div>
136
137             <div>Planted: </div>
138             <div>5/25/24 </div>
139
140             <div>Harvest: </div>
141             <div>6/7/24 </div>
142
143           </div>
144
145           <div class = "plantInfo">
146
147             <div>
148                 <img src = "{{ url_for('static', filename='images/
                        plants/romaine.jpeg')}}" alt = "Lettuce" height =
                        90% width = 90% class = "plantImages">
149             </div>
150
151             <div> Lettuce</div>
152
153             <div>Container: </div>
154             <div> B </div>
155
156             <div>Planted: </div>
157             <div>5/25/24 </div>
158
159             <div>Harvest: </div>
160             <div>6/7/24 </div>
161
162           </div>
163
164         </div>
165         </article>
166
167     </body>
168     </html>
```

Listing 20: Plant Page HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">

<title>LuLi Hydroponics</title>
<link href="{{ url_for('static', filename='style.css') }}" rel
   ="stylesheet">
<script src="scripts/settingsSlide.js"></script>

</head>
<body>
<header>
    <img class="logo" src="{{ url_for('static', filename='
       images/hydroponic.png')}}" alt="hydroPlant" width="50"
       height="50" style="float: left;">
    <img src="{{ url_for('static', filename='images/luli_logo.
       png')}}" alt="Luli Hydroponics" width="60" height="60"
       style="float: left;">
    <div class="corner_topRight"></div>
</header>
<nav class = "nav_list">

    <div class = "button" >
       <a href="/" target=" _self"><img src = "{{ url_for('
          static', filename='images/user.png')}}" alt = "login
          " height= 50 width = 50></a>
       <label class="button_label"> Login </label>
    </div>
    <div class = "button" >
       <a href="/plants" target=" _self"> <img src = "{{
          url_for('static', filename='images/pot.png')}}" alt
          = "plant" height= 50 width = 50> </a>
       <label class="button_label"> Plants </label>
    </div>
    <div class = "button" >
       <a href="/friends" target=" _self"> <img src = "{{
          url_for('static', filename='images/friends.png')}}"
          alt = "plant" height= 50 width=50> </a>
       <label class="button_label"> Friends </label>
    </div>
    <div class = "button" >
       <a href="/forum" target=" _self"> <img src = "{{
          url_for('static', filename='images/discussion.png')
          }}" alt = "plant" height= 50 width=50> </a>
       <label class="button_label"> Forum </label>
    </div>
    <div class = "button" >
       <a href="/settings" target=" _self"> <img src = "{{
          url_for('static', filename='images/setting.png')}}"
```

```
            alt = "plant" height= 50 width=50> </a>
37          <label class="button_label"> Settings </label>
38       </div>
39          <div class = "corner_bottomLeft"> </div>
40       </nav>
41    <article>
42
43       <form id="settingsForm" action="/save_settings" method="
            POST">
44          <div class="settingsContainer">
45
46             <label for="water_interval" id="
                water_intervalLabel">Water plants every <span
                id="water_intervalValue">{{ settings.get('
                water_interval', 12) }} hours</span></label>
47             <div class="sliderContainer">
48                <input type="range" id="water_interval" name="
                   water_interval" min="1" max="24" value="{{
                   settings.get('water_interval', 12) }}"
                   oninput="document.getElementById('
                   water_intervalValue').textContent = this.
                   value + ' hours'">
49             </div>
50
51
52             <label for="led_duration" id="led_durationLabel">
                Turn on LEDs for <span id="led_durationValue">
                {{ settings.led_duration if settings.
                led_duration else '12' }}</span> hours</label>
53             <div class="sliderContainer">
54                <input type="range" id="led_duration" name="
                   led_duration" min="1" max="24" value="{{
                   settings.led_duration if settings.
                   led_duration else '12' }}" oninput="
                   document.getElementById('led_durationValue
                   ').textContent = this.value">
55             </div>
56
57             <label for="pump_duration" id="pump_durationLabel"
                >Water for <span id="pump_durationValue">{{
                settings.pump_duration if settings.
                pump_duration else '5' }}</span> minutes per
                cycle</label>
58
59             <div class="sliderContainer">
60                <input type="range" id="pump_duration" name="
                   pump_duration" min="1" max="15" value="{{
                   settings.pump_duration if settings.
                   pump_duration else '5' }}" oninput="
                   document.getElementById('pump_durationValue
                   ').textContent = this.value">
```

```
61                </div>
62
63                <div class="settingsButton">
64                    <button type="submit" id="saveSettingsButton">
                          Save Settings</button>
65                </div>
66            </div>
67        </form>
68
69        <div class = "plantInfoSettingsContainer">
70
71            <div id = "plantAmount" class = "plantAmount" >
72                <label for="quantity">Number of Plants:</label>
73                <input type="number" id="quantity" name="quantity"
                      min="1" max="6" class = "spinVisible" onchange
                      = "updateSettingsCarousel()">
74            </div>
75
76            <div id = "plantCarousel" class = "plantCarousel">
77                <div id="indPlantSettings" class = "
                     indPlantSettings" >
78
79                    <div>
80                        <label for="plantType">Plant Type:</label>
81                        <select id="plantType" name="plantType">
82                                <option value="Spinach">Spinach</
                                    option>
83                                <option value="Romaine">Romaine</
                                    option>
84                                <option value="Basil">Basil</
                                    option>
85                                <option value="Cilantro">Cilantro<
                                    /option>
86                                <option value="Green Onions">Green
                                     Onions</option>
87                                <option value="Kale">Kale</option>
88                                <option value="Mint">Mint</option>
89                                <option value="Oregano">Oregano</
                                    option>
90                                <option value="Parsley">Parsley</
                                    option>
91                                <option value="Radish">Radish</
                                    option>
92
93                        </select>
94                    </div>
95
96                    <div>
97                        <label for="start">Date Planted:</label>
98                        <input type="date" id="start" name="trip-
                              start" value="2024-01-01" min="
```

```
                        2023-01-01" max="2026-12-31" class="
                        custom-calendar"/>
                </div>

                <div>
                    <label for="start">Date of Harvest:</label
                        >
                    <input type="date" id="start" name="trip-
                        start" value="2024-01-01" min="
                        2023-01-01" max="2026-12-31" />
                </div>

                <div>
                    <label for="letters">Choose Container:</
                        label>
                    <select id="letters" name="letters">
                            <option value="A">A</option>
                            <option value="B">B</option>
                            <option value="C">C</option>
                    </select>
                </div>
            </div>
        </div>

    </div>

</article>
<script src="scripts/settingsSlide.js"  defer></script>
<script src="scripts/dynamicSettingsCarousel.js"  defer></
    script>
</body>
</html>
```

Listing 21: Settings Page HTML

```
1
2       .nav_list
3       {
4           display: flex;
5           flex-direction: column;
6           justify-content: center;
7           align-items: center;
8       }
9
10
11
12      .button
13      {
14          display: flex;
15          flex-direction: column;
16          align-items: center;
17          padding: 10px 10px ;
18          border: 3px solid black;
19          width: 70px;
20          height: 70px;
21          background: rgb(206 249 173);
22          box-shadow: 4px 4px 4px black;
23          border-radius: 15px 15px;
24          margin-top: auto;
25      }
26
27      .user
28      {
29          display: inline-block;
30          width: 150px;
31          height: 150px;
32          border-radius: 15px 15px 15px 15px;
33          margin-top: 40px;
34          margin-bottom: 60px;
35          background-repeat: no-repeat;
36          background-position: center center;
37          background-size: cover;
38          background-image: url('hydroponic.png');
39      }
40
41      .profile
42      {
43          margin-right: 04px;
44      }
45
46      .password
47      {
48          margin-right: 4px;
49      }
50
```

```
51
52      .createAccount
53      {
54          display:grid;
55          grid-template-columns: 40% 60%;
56          width: 80%;
57          height: 80%;
58          background: rgb(206 249 173);
59          box-shadow: 4px 4px 4px black;
60          border: 4px solid black;
61          border-radius: 15px 15px 15px 15px;
62          justify-content: center;
63          align-items: center;
64
65      }
66
67      .welcomeMessage
68      {
69          display:flex;
70          background-color:rgb(237 176 102);
71          width: 100%;
72          height: 25%;
73          border-radius: 0px 11px 0px 0px;
74          justify-content: center;
75          align-items: center;
76          margin-top: -41px;
77
78      }
79
80      .createAccountForm
81      {
82          display: flex;
83          flex-direction: column;
84          gap: 40px;
85          height: 100%;
86          width: 100%;
87          justify-content: center;
88          align-items: center;
89
90      }
91
92      .createAcountButton
93      {
94      width: 100px;
95      }
96
97
98
99      .circleLogo
100     {
101         display: flex;
```

```
102         flex-direction: column;
103         background-color: rgb(126 198 204);
104         border-radius: 12px 0px 0px 12px;
105         width: 100%;
106         height: 100%;
107         justify-content: center;
108         align-items: center;
109         gap: 20px;
110     }
111 
112 
113     .logoFrontPage
114     {
115         background-color: rgb(132 158 214);
116         box-shadow: 4px 4px 4px black;
117         border: 4px solid black;
118         border-radius: 50%;
119         padding: 30px;
120     }
121 
122 
123     .module_login
124     {
125         font-size: 23px;
126         display: flex;
127         gap:15px;
128         flex-direction:column;
129         align-items: center;
130         width: 400px;
131         height: 500px;
132         background: rgb(206 249 173);
133         box-shadow: 4px 4px 4px black;
134         border: 4px solid black;
135         border-radius: 15px 15px 15px 15px;
136     }
137 
138     .module_login div
139     {
140         display: flex;
141         flex-direction: row;
142     }
143 
144      .button_label
145     {
146         font-size: 18px;
147     }
148 
149     .button:hover
150     {
151         background: gainsboro;
152         transform: scale(1.05, 1.05);
```

```
153     }
154
155     body
156     {
157
158         background: rgba(206, 249, 173, 0.578);
159         display: grid;
160         grid-template-columns: 130px 1fr;
161         grid-template-rows:auto 670px;
162         grid-template-areas:
163         "header header "
164         "nav art ";
165         font-size: 2em;
166         text-align: center;
167         border-radius:15px;
168         box-shadow: 4px 4px 4px black;
169         min-width: 700px;
170         font-family: monospace;
171     }
172
173
174     header
175     {
176         grid-area:header;
177         background: rgb(126 198 204);
178         line-height: 100%;
179         border-radius:15px 15px 0px 0px;
180         margin-bottom: 2px;
181         border: 4px solid black;
182     }
183
184     .corner_topRight
185     {
186         background: rgb(206 249 173);
187         line-height: 100%;
188         width: 10%;
189         height: 100%;
190         border-left: 4px solid black;
191         border-radius: 0px 11px 0px 0px;
192         float:right;
193     }
194
195     .corner_bottomLeft
196     {
197         background: rgb(206 249 173);
198         width:100%;
199         height:20%;
200         border-top: 4px solid black;
201         border-radius: 0px 0px 0px 10px;
202         margin-top: auto;
203     }
```

```
204
205     nav
206     {
207         grid-area:nav;
208         background: rgb(132 158 214);
209         border: 4px solid black;
210         border-radius: 0px 0px 0px 15px;
211         margin-top: -4px;
212         margin-right: -4px;
213     }
214
215     article
216     {
217         grid-area: art;
218         display: flex;
219         background: rgb(237 176 102);
220         border-radius: 0px 0px 15px 0px;
221         width: auto;
222         border: 4px solid black;
223         margin-top: -4px;
224         justify-content: center;
225         align-items: center;
226
227     }
228
229
230     .measurements
231     {
232         font-size: 23px;
233         display: grid;
234         grid-template-columns: 20% 20% 20% 20% 20% ;
235         grid-template-rows: 85% 7.5% 7.5%;
236         width: 500px;
237         height: 500px;
238         background: rgb(206 249 173);
239         box-shadow: 4px 4px 4px black;
240         border: 4px solid black;
241         border-radius: 15px 15px 15px 15px;
242         place-items: center;
243         margin: auto 2% auto 2%
244
245     }
246
247     .plantSection
248     {
249         font-size: 23px;
250         display: flex;
251         scroll-snap-type: x mandatory;
252         background-color: rgb(206 249 173);
253         overflow-x: auto;
254         width: 450px;
```

```
255         height: 500px;
256         border-radius: 15px 15px 15px 15px;
257         place-items: center;
258         max-width: 350;
259         position: relative;
260         box-shadow: 4px 4px 4px black;
261         border: 4px solid black;
262         overflow: auto;
263     }
264 
265 
266 
267     .plantInfo
268     {
269         font-size: 23px;
270         display: grid;
271         flex-shrink: 0;
272         scroll-snap-align: start;
273         scroll-behavior: smooth;
274         scroll-snap-type:proximity;
275         grid-template-columns: 50% 50% ;
276         grid-template-rows: 30% 17.5% 17.5% 17.5% 17.5%;
277         width: 450px;
278         height: 100%;
279         background: rgb(206 249 173);
280         place-items: center;
281         scroll-snap-align: center;
282 
283 
284     }
285 
286     .plantImages
287     {
288     margin: 15px auto auto 15px;
289     border-radius:15px;
290     height: 50%;
291     box-shadow: 2px 2px 2px black;
292     }
293 
294     /* width */
295     ::-webkit-scrollbar {
296         width: 2px;
297         height: 20px;
298         border-radius: 15px;
299      }
300 
301      /* Track */
302      ::-webkit-scrollbar-track {
303         background: rgb(132 158 214);
304         border: 4px 0px 0px 0px solid black;
305         border-radius: 0px 0px 10px 10px  ;
```

```
306       }
307
308       /* Handle */
309       ::-webkit-scrollbar-thumb {
310         background: rgb(126 198 204);
311         border-radius: 15px;
312       }
313
314       /* Handle on hover */
315       ::-webkit-scrollbar-thumb:hover {
316         background: rgb(237 176 102);
317       }
318
319       .measurementBar
320       {
321          position: relative;
322          height: 380px;
323          width : 35px;
324          border-radius: 15px 15px 15px 15px;
325          border: 4px solid black;
326          background: white;
327          margin: auto auto auto auto;
328       }
329
330       .barProgress {
331          position: absolute;
332          bottom: 0;
333          border-radius: 0px 0px 11px 11px;
334          margin: auto;
335       }
336
337
338       .forumContainer
339       {
340          display:grid;
341          grid-template-columns: 80% 20%;
342          grid-area:
343          "forumPost" "forumMenu" ;
344          overflow: auto;
345          width: 100%;
346          height: 100%;
347          place-items: center;
348          justify-content: center;
349          overflow: auto;
350       }
351
352       .forumPosts
353       {
354          display: flex;
355          flex-direction: column;
356          gap:15px;
```

```
357         height: 100%;
358         width: 90%;
359 
360     }
361 
362     .post
363     {
364 
365         width: 100%;
366         height: 400px;
367         border: 4px solid black;
368         box-shadow:  2px 2px 2px black;
369         background-color: rgb(132 158 214);
370         text-align: left;
371     }
372 
373     .forumMenu
374     {
375         position: fixed;
376         bottom: 100px; /* Adjust as needed */
377         right: 40px; /* Adjust as needed */
378         display: flex;
379         flex-direction: column;
380         height: 200px;
381         width : 200px;
382         justify-content:end;
383 
384 
385     }
386 
387     .forumButton
388     {
389         width:40% ;
390         height:30%;
391         background-color: rgb(206 249 173);
392         border-radius: 15px;
393         border: 4px solid black;
394         box-shadow:  2px 2px 2px black;
395     }
396 
397     .forumButton:hover
398     {
399         transform: scale(1.1);
400     }
401 
402 
403     .settingsContainer
404     {
405         font-size: 23px;
406         display: grid;
407         background-color: rgb(132 158 214) ;
```

```
        grid-template-columns: 20% 80%;
        grid-template-rows: 25% 25% 25% 25% ;
        grid-area:
        "waterCycle" "waterCycle"
        "waterAmount" "waterAmount"
        "Lightcycle" "Lightcycle"
        "LightDuration" "LightDuration";
        place-items: center;
        width: 45%;
        height: 80%;
        border-radius: 15px 15px 15px 15px;
        border: 4px solid black;
        box-shadow:  2px 2px 2px black;
        margin: auto;
    }

    .plantInfoSettingsContainer
    {
        font-size: 23px;
        display: grid;
        background-color: rgb(206 249 173) ;
        grid-template-rows: 20% 80%;
        place-items: center;
        width: 30%;
        height: 80%;
        border-radius: 15px 15px 15px 15px;
        border: 4px solid black;
        box-shadow:  2px 2px 2px black;
        margin: auto;
    }

    .plantAmount
    {
        display: flex;
        width: 100%;
        height:100%;
        background-color: rgb(132 158 214);
        border-radius: 12px 12px 0px 0px;
        flex-direction: row;
        gap: 10px;
        place-items: center;
        justify-content: center;
    }

    .indPlantSettings
    {
        display: flex;
        flex-direction: column;
        gap: 15px;
        width:100%;
        height:100%;
```

```
459         flex-shrink: 0;
460         scroll-snap-align: start;
461         scroll-behavior: smooth;
462         scroll-snap-type:proximity;
463         scroll-snap-align: center;
464         place-items: center;
465         justify-content: center;
466 
467     }
468 
469     .plantCarousel
470     {
471         display: flex;
472         flex-direction: row;
473         height:100%;
474         width:100%;
475         font-size: 23px;
476         scroll-snap-align: start;
477         scroll-behavior: smooth;
478         scroll-snap-type:proximity;
479         scroll-snap-align: center;
480         overflow: auto;
481         scroll-snap-type: x mandatory;
482 
483     }
484     .friendList {
485         display: flex;
486         flex-direction: row;
487         scroll-snap-align: start;
488         scroll-behavior: smooth;
489         scroll-snap-type: x mandatory;
490         overflow: auto;
491         justify-content:start;
492     }
493 
494     .friendModule
495     {
496         font-size: 23px;
497         display: grid;
498         grid-template-rows: 50% 50%;
499         grid-template-columns: 100%;
500         width: 500px;
501         height: 500px;
502         background: rgb(206 249 173);
503         box-shadow: 4px 4px 4px black;
504         border: 4px solid black;
505         border-radius: 15px 15px 15px 15px;
506         flex-shrink: 0;
507         justify-content: start;
508         margin: 10px;
509     }
```

```
510
511     .userProfile
512     {
513        display: grid;
514        grid-template-columns: 50% 50%;
515        justify-content: center;
516        align-items: center;
517        background-color: rgb(132 158 214);
518        width: 100%;
519        height: 100%;
520        border-radius: 11px 11px 0px 0px;
521
522     }
523
524     .profileImage
525     {
526           width: 100%;
527           height: 80%;
528           border-radius: 15px;
529           background-color: #ccc;
530           background-size: cover;
531           background-position: center;
532           margin:10px;
533           box-shadow: 4px 4px 4px black;
534           border: 2px solid black;
535     }
536
537     .userInfo
538     {
539        display:grid;
540        grid-template-columns: 80% 20%;
541        height: 100%;
542        width: 100%;
543
544     }
545
546     .userAbout
547     {
548        text-align: left;
549        margin:10px
550     }
551
552
553     .userInteraction
554     {
555        background-color: rgb(126 198 204);
556        width: 100%;
557        height: 100%;
558        border-radius: 0px 0px 11px 0px;
559     }
560
```

```
561
562     .downloadSettings{
563        background-color: rgb(237 176 102);
564        width: 80%;
565        height: 30%;
566        border-radius: 15px;
567        box-shadow: 4px 4px 4px black;
568        border: 2px solid black;
569        margin: 10px;
570        cursor: pointer;
571        transition: transform 0.2s;
572     }
573
574     .downloadSettings:hover {
575        transform: scale(1.1);
576     }
577
578
579     .spinVisible::-webkit-outer-spin-button,
580     .spinVisible::-webkit-inner-spin-button
581     {
582         opacity: 1;
583     }
584
585     /* width */
586     input[type="range"] {
587        width: 85%;
588        height: 5px;
589        -webkit-appearance:none;
590        appearance: none;
591        border: 3px solid rgb(206 249 173);
592        background-color: rgb(206 249 173);
593        box-shadow: 3px 3px 3px black;
594        /* Add rounded corners */
595        border-radius: 20px;
596      }
597
598      input[type="range"]::-webkit-slider-thumb
599      {
600        width: 20px;
601        height: 30px;
602        background: rgb(237 176 102);
603        border-radius: 25%;
604        border: 2px solid black;
605        box-shadow: 1px 1px 1px black;
606        -webkit-appearance: none;
607
608      }
609
610      .settingsContainer input[type="range"]:active::-webkit-slider
          -thumb {
```

```
611         transform: scale(1.3);
612     }
613 
614     .settingInput
615     {
616         width: auto;
617         height: auto;
618         display: flex;
619         flex-direction: row;
620         gap: 10px;
621     }
622 
623     #slideLabel
624     {
625         display:block;
626         width: 100%;
627         height: 100%;
628         background-color: rgb(126 198 204);
629         border-radius: 0px 0px 0px 0px;
630         align-content: center;
631 
632         margin-left:auto;
633         font-size: 20px;
634     }
635 
636     #slideLabelTop
637     {
638         display:block;
639         width: 100%;
640         height: 100%;
641         background-color: rgb(126 198 204);
642         border-radius: 12px 0px 0px 0px;
643         align-content: center;
644         margin-left:auto;
645         font-size: 20px;
646     }
647     #containerSlideValue
648     {
649 
650         display: flex;
651         flex-direction: row;
652         gap: 15px;
653         margin-left: 5px;
654     }
655 
656     #slideValue
657     {
658         font-size: 18px;
659         margin: auto;
660     }
661 
```

```
662
663
664     #slideLabelBottom
665     {
666         display:block;
667         width: 100%;
668         height: 100%;
669         background-color: rgb(126 198 204);
670         border-radius: 0px 0px 0px 12px;
671         align-content: center;
672         margin-left:auto;
673         font-size: 20px;
674     }
675
676     .type
677     {
678         font-size: 20px;
679         font-family: monospace;
680         margin: auto auto 80% auto;
681     }
682
683     aside
684     {
685         background: goldenrod;
686         line-height: 180px;
687     }
688
689     section
690     {
691         background: lightsteelblue;
692         line-height: 90px;
693     }
694
695     footer
696     {
697         background: lemonchiffon;
698         line-height: 40px;
699     }
700
701     .logo
702     {
703         margin-top: 5px;
704         margin-left: 10px;
705         margin-right: 10px;
706     }
```

Listing 22: CSS Stlye Sheet

B Hardware Appendix

B.1 Datasheet: Raspberry Pi Pico W

Raspberry Pi Pico W Datasheet

An RP2040-based microcontroller board with wireless

Colophon

build-date: 2024-02-02
build-version: 169135e-dirty

About the SDK

Throughout the text "the SDK" refers to our Raspberry Pi Pico SDK. More details about the SDK can be found in the **Raspberry Pi Pico C/C++ SDK** book. Source code included in the documentation is Copyright © 2020-2023 Raspberry Pi Ltd (formerly Raspberry Pi (Trading) Ltd.) and licensed under the 3-Clause BSD license.

Legal disclaimer notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (“RESOURCES”) ARE PROVIDED BY RASPBERRY PI LTD (“RPL”) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage (“High Risk Activities”). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL’s Standard Terms. RPL’s provision of the RESOURCES does not expand or otherwise modify RPL’s Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Table of contents

Chapter 1. About Raspberry Pi Pico W

Raspberry Pi Pico W is a microcontroller board based on the Raspberry Pi RP2040 microcontroller chip.

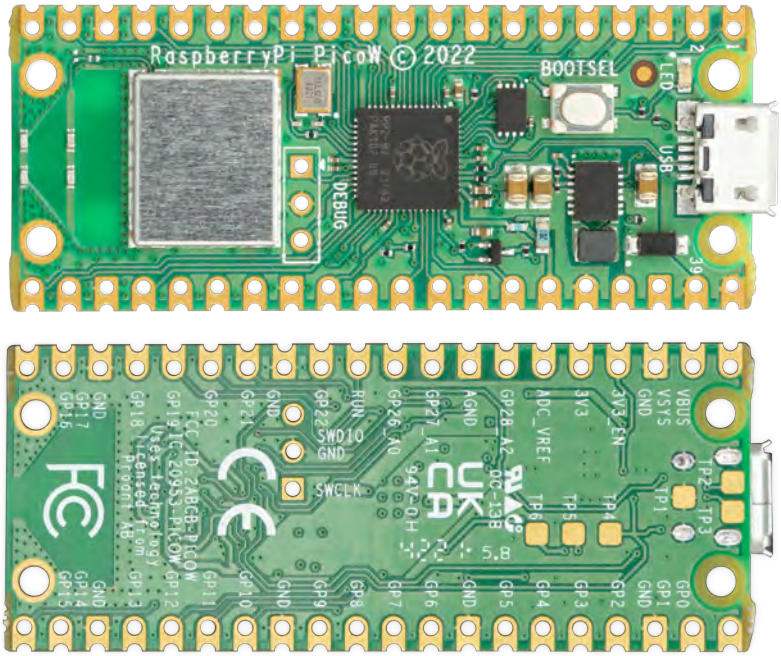


Figure 1. The Raspberry Pi Pico W Rev3 board.

Raspberry Pi Pico W has been designed to be a low cost yet flexible development platform for RP2040, with a 2.4GHz wireless interface and the following key features:

- RP2040 microcontroller with 2MB of flash memory
- On-board single-band 2.4GHz wireless interfaces (802.11n, Bluetooth 5.2)
  - Support for Bluetooth LE Central and Peripheral roles
  - Support for Bluetooth Classic
- Micro USB B port for power and data (and for reprogramming the flash)
- 40-pin 21mm×51mm 'DIP' style 1mm thick PCB with 0.1" through-hole pins also with edge castellations
  - Exposes 26 multi-function 3.3V general purpose I/O (GPIO)
  - 23 GPIO are digital-only, with three also being ADC capable
  - Can be surface-mounted as a module
- 3-pin Arm serial wire debug (SWD) port
- Simple yet highly flexible power supply architecture
  - Various options for easily powering the unit from micro USB, external supplies or batteries
- High quality, low cost, high availability
- Comprehensive SDK, software examples and documentation

For full details of the RP2040 microcontroller please see the **RP2040 Datasheet** book. Key features include:

- Dual-core cortex M0+ at up to 133MHz

  - On-chip PLL allows variable core frequency
- 264kB multi-bank high performance SRAM
- External Quad-SPI flash with eXecute In Place (XIP) and 16kB on-chip cache
- High performance full-crossbar bus fabric
- On-board USB1.1 (device or host)
- 30 multi-function general purpose I/O (four can be used for ADC)
  - 1.8-3.3V I/O voltage
- 12-bit 500ksps analogue to digital converter (ADC)
- Various digital peripherals
  - 2 × UART, 2 × I2C, 2 × SPI, 16 × PWM channels
  - 1 × timer with 4 alarms, 1 × real time clock
- 2 × programmable I/O (PIO) blocks, 8 state machines in total
  - Flexible, user-programmable high-speed I/O
  - Can emulate interfaces such as SD card and VGA

NOTE

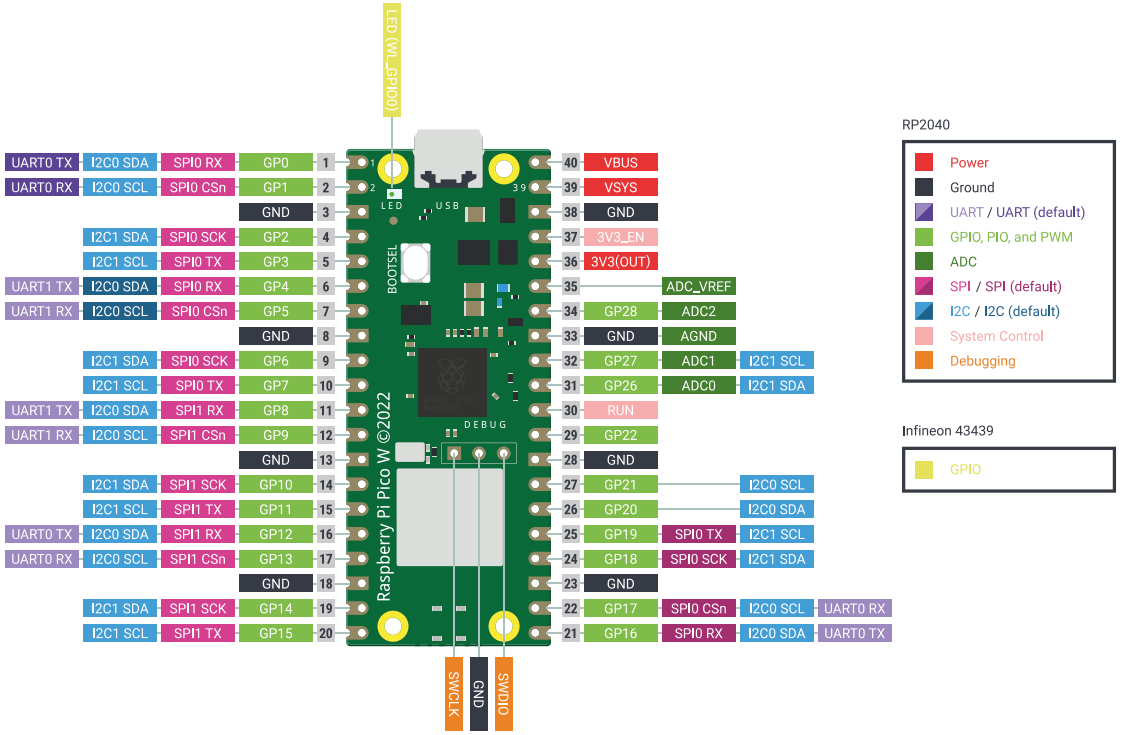
Raspberry Pi Pico W I/O voltage is fixed at 3.3V

Raspberry Pi Pico W provides a minimal yet flexible external circuitry to support the RP2040 chip: flash memory (Winbond W25Q16JV), crystal (Abracon ABM8-272-T3), power supplies and decoupling, and USB connector. The majority of the RP2040 microcontroller pins are brought to the user I/O pins on the left and right edge of the board. Four RP2040 I/O are used for internal functions: driving an LED, on-board switch mode power supply (SMPS) power control, and sensing the system voltages.

Pico W has an on-board 2.4GHz wireless interface using an Infineon CYW43439. The antenna is an onboard antenna licensed from ABRACON (formerly ProAnt). The wireless interface is connected via SPI to the RP2040.

Pico W has been designed to use either soldered 0.1-inch pin-headers (it is one 0.1-inch pitch wider than a standard 40-pin DIP package), or to be positioned as a surface-mountable 'module', as the user I/O pins are also castellated. There are SMT pads underneath the USB connector and BOOTSEL button, which allow these signals to be accessed if used as a reflow-soldered SMT module.

Figure 2. The pinout of the Pico W Rev3 board



Raspberry Pi Pico W uses an on-board buck-boost SMPS which is able to generate the required 3.3V (to power RP2040 and external circuitry) from a wide range of input voltages (~1.8 to 5.5V). This allows significant flexibility in powering the unit from various sources, such as a single lithium-ion cell, or three AA cells in series. Battery chargers can also be very easily integrated with the Pico W powerchain.

Reprogramming the Pico W flash can be done using USB (simply drag and drop a file onto the Pico W, which appears as a mass storage device), or the standard serial wire debug (SWD) port can reset the system and load and run code without any button presses. The SWD port can also be used to interactively debug code running on the RP2040.

Getting started with Pico W

The **Getting started with Raspberry Pi Pico** book walks through loading programs onto the board, and shows how to install the C/C++ SDK and build the example C programs. See the **Raspberry Pi Pico Python SDK** book to get started with MicroPython, which is the fastest way to get code running on Pico W.

1.1. Raspberry Pi Pico W design files

The source design files, including the schematic and PCB layout, are made available openly except for the antenna. The Niche™ antenna is an Abracon/Proant patented antenna technology. Please contact niche@abracon.com for information on licensing.

Schematic The schematic is reproduced in Appendix B. The schematic is also distributed alongside the layout files.

Layout The CAD files, including PCB layout, can be found here. Note that Pico W was designed in Cadence Allegro PCB Editor, and opening in other PCB CAD packages will require an import script or plugin.

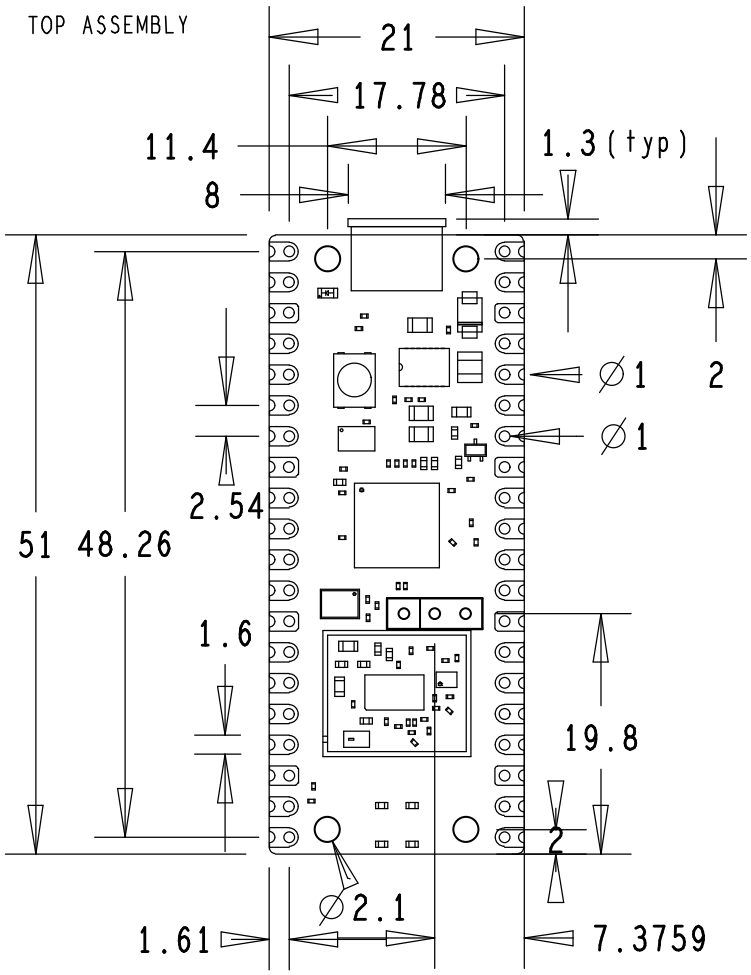
STEP 3D A STEP 3D model of Raspberry Pi Pico W, for 3D visualisation and fit-check of designs which include Pico W as a module, can be found here.

Fritzing A Fritzing part for use in e.g. breadboard layouts can be found here.

Chapter 2. Mechanical specification

The Pico W is a single sided 51mm × 21mm × 1mm PCB with a micro USB port overhanging the top edge, and dual castellated/through-hole pins around the two long edges. The onboard wireless antenna is located on the bottom edge. To avoid detuning the antenna, no material should intrude into this space. Pico W is designed to be usable as a surface-mount module as well as presenting a dual inline package (DIP) format, with the 40 main user pins on a 2.54mm (0.1") pitch grid with 1mm holes, compatible with veroboard and breadboard. Pico W also has four 2.1mm (± 0.05mm) drilled mounting holes to provide for mechanical fixing(see Figure 3).

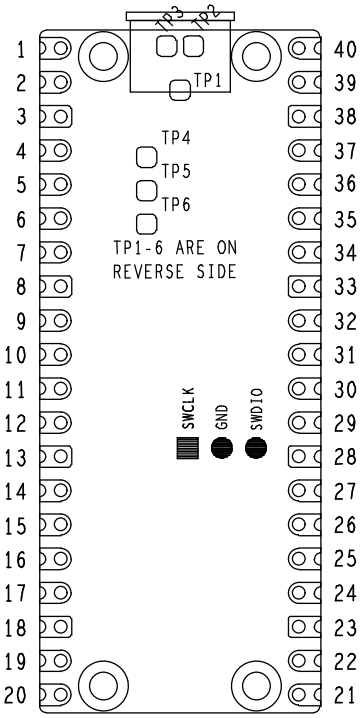
Figure 3. The dimensions of the Pico W



2.1. Pico W pinout

The Pico W pinout has been designed to directly bring out as much of the RP2040 GPIO and internal circuitry function as possible, while also providing a suitable number of ground pins to reduce electro-magnetic interference (EMI) and signal crosstalk. RP2040 is built on a modern 40nm silicon process, so its digital I/O edge rates are very fast.

Figure 4. The pin numbering of the Pico W



NOTE

The physical pin numbering is shown in Figure 4. For pin allocation see Figure 2, or the full Pico W schematics in Appendix B.

A few RP2040 GPIO pins are used for internal board functions:

GPIO29	OP/IP wireless SPI CLK/ADC mode (ADC3) to measure VSYS/3
GPIO25	OP wireless SPI CS - when high also enables GPIO29 ADC pin to read VSYS
GPIO24	OP/IP wireless SPI data/IRQ
GPIO23	OP wireless power on signal
WL_GPIO2	IP VBUS sense - high if VBUS is present, else low
WL_GPIO1	OP controls the on-board SMPS power save pin (Section 3.4)
WL_GPIO0	OP connected to user LED

Apart from GPIO and ground pins, there are seven other pins on the main 40-pin interface:

PIN40	VBUS
PIN39	VSYS
PIN37	3V3_EN
PIN36	3V3
PIN35	ADC_VREF
PIN33	AGND

PIN30 RUN

VBUS is the micro-USB input voltage, connected to micro-USB port pin 1. This is nominally 5V (or 0V if the USB is not connected or not powered).

VSYS is the main system input voltage, which can vary in the allowed range 1.8V to 5.5V, and is used by the on-board SMPS to generate the 3.3V for the RP2040 and its GPIO.

3V3_EN connects to the on-board SMPS enable pin, and is pulled high (to VSYS) via a 100kΩ resistor. To disable the 3.3V (which also de-powers the RP2040), short this pin low.

3V3 is the main 3.3V supply to RP2040 and its I/O, generated by the on-board SMPS. This pin can be used to power external circuitry (maximum output current will depend on RP2040 load and VSYS voltage; it is recommended to keep the load on this pin under 300mA).

ADC_VREF is the ADC power supply (and reference) voltage, and is generated on Pico W by filtering the 3.3V supply. This pin can be used with an external reference if better ADC performance is required.

AGND is the ground reference for GPIO26-29. There is a separate analogue ground plane running under these signals and terminating at this pin. If the ADC is not used or ADC performance is not critical, this pin can be connected to digital ground.

RUN is the RP2040 enable pin, and has an internal (on-chip) pull-up resistor to 3.3V of about ~50kΩ. To reset RP2040, short this pin low.

Finally, there are also six test points (TP1-TP6), which can be accessed if required, for example if using as a surface-mount module. These are:

TP1 Ground (close-coupled ground for differential USB signals)

TP2 USB DM

TP3 USB DP

TP4 WL_GPIO1/SMPS PS pin (do not use)

TP5 WL_GPIO0/LED (not recommended to be used)

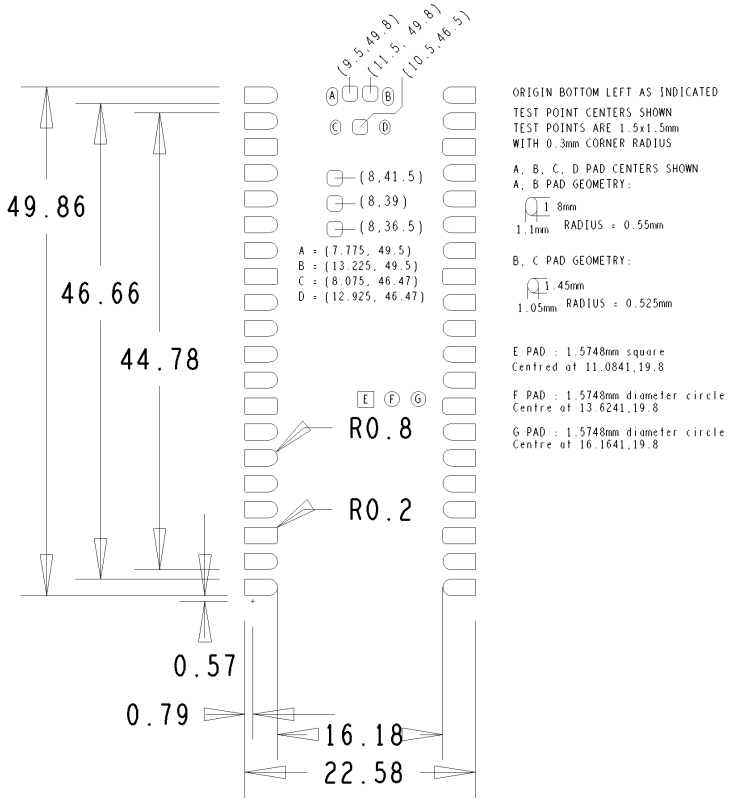
TP6 BOOTSEL

TP1, TP2 and TP3 can be used to access USB signals instead of using the micro-USB port. TP6 can be used to drive the system into mass-storage USB programming mode (by shorting it low at power-up). Note that TP4 is not intended to be used externally, and TP5 is not really recommended to be used as it will only swing from 0V to the LED forward voltage (and hence can only really be used as an output with special care).

2.2. Surface-mount footprint

The following footprint (Figure 5) is recommended for systems which will be reflow-soldering Pico W units as modules.

Figure 5. The SMT footprint of the Pico W Rev3 board.

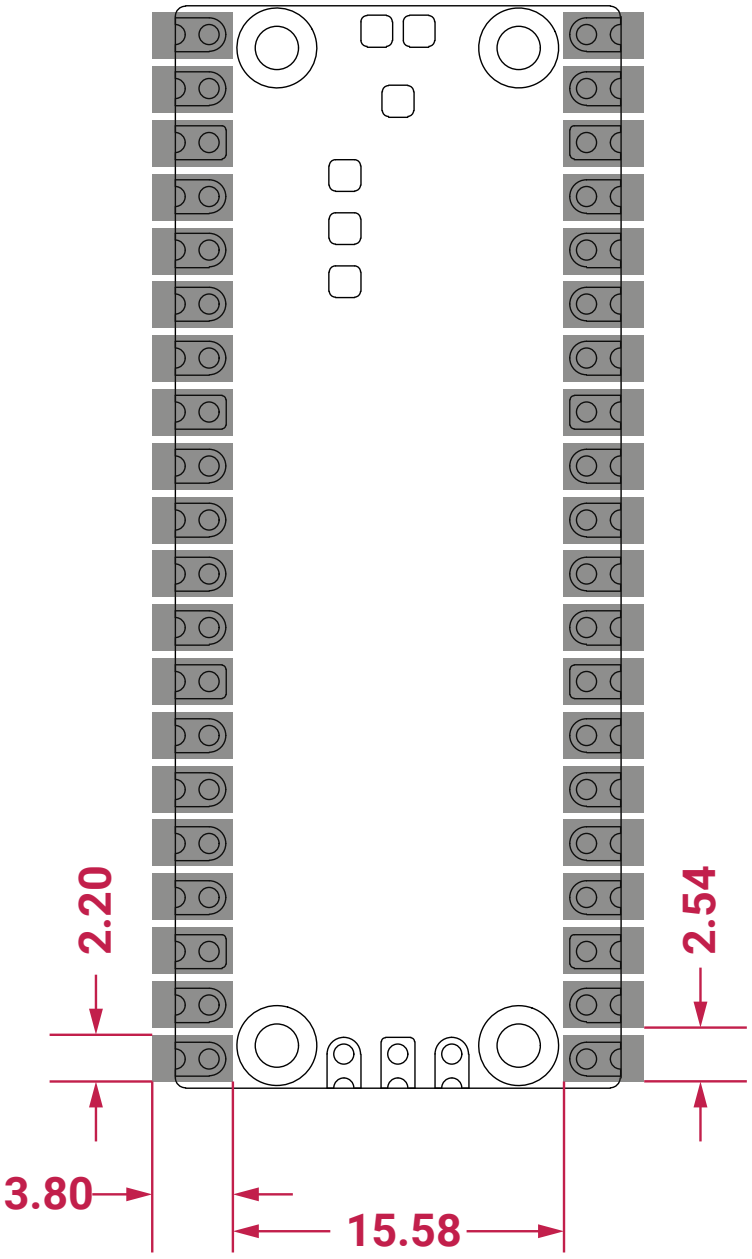


The footprint shows the test point locations and pad sizes as well as the 4 USB connector shell ground pads (A,B,C,D). The USB connector on Pico W is a through-hole part, which provides it with mechanical strength. The USB socket pins do not protrude all the way through the board, however solder does pool at these pads during manufacture and can stop the module sitting completely flat. Hence we provide pads on the SMT module footprint to allow this solder to reflow in a controlled manner when Pico W goes through reflow again.

For test points that are not used, it is acceptable to void any copper under these (with suitable clearance) on the carrier board.

Through trials with customers, we have determined that the paste stencil must be bigger than the footprint. Over-pasting the pads ensures the best possible results when soldering. The following paste stencil (Figure 6) indicates the dimensions of solder paste zones on the Pico W. We recommend paste zones 163% larger than the footprint.

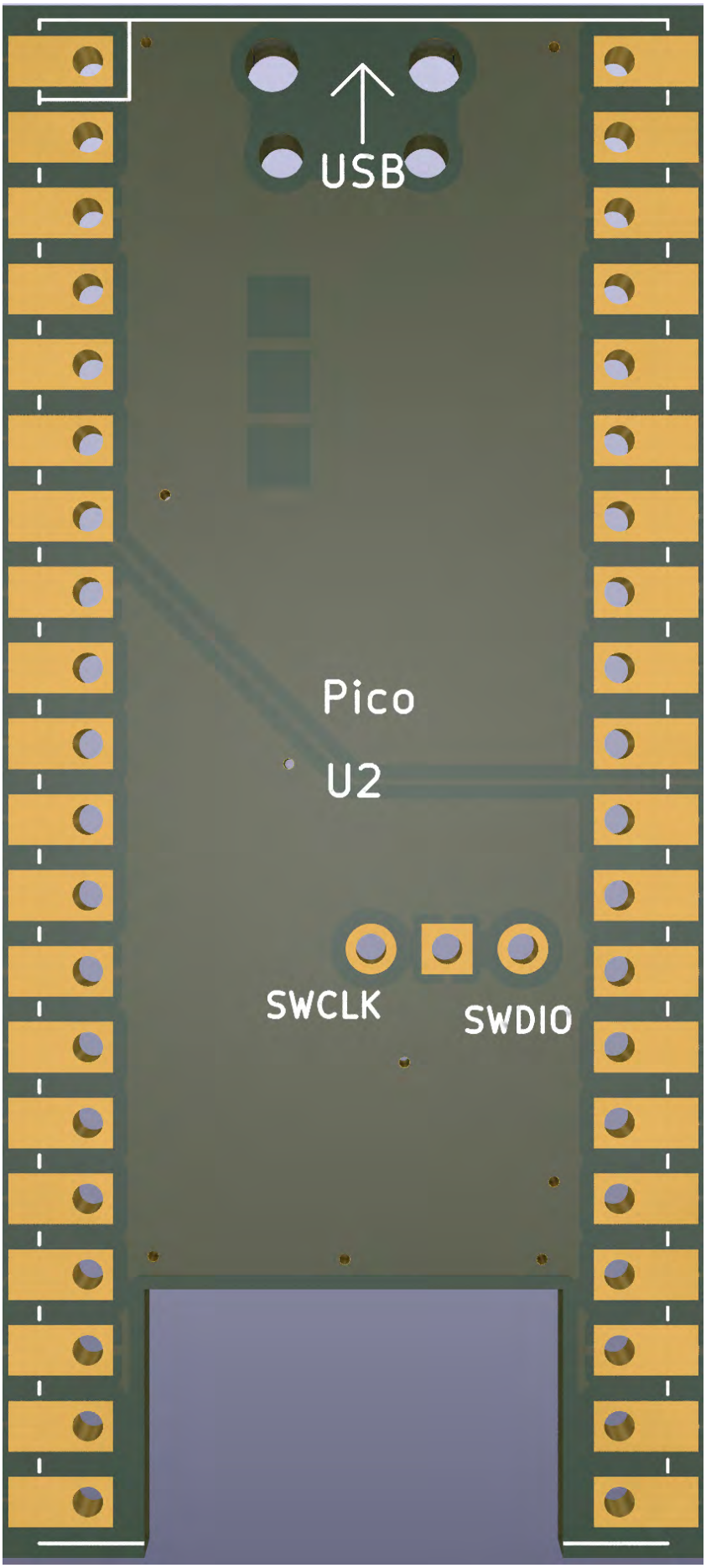
Figure 6. The paste stencil of the Raspberry Pi Pico W Rev3 board.



2.2.1. Keep-out area

There is a cutout for the antenna (14mm × 9mm). If anything is placed close to the antenna (in any dimension) the effectiveness of the antenna is reduced. Raspberry Pi Pico W should be placed on the edge of a board and not enclosed in metal to avoid creating a Faraday cage. Adding ground to the sides of the antenna improves the performance slightly.

Figure 7. Surface-mount and throughhole footprint for attaching to Raspberry Pi Pico W



2.3. Recommended operating conditions

Operating conditions for the Pico W are largely a function of the operating conditions specified by its components.

Operating Temp Max	70°C (including self-heating)
Operating Temp Min	-20°C
VBUS	5V ± 10%.
VSYS Min	1.8V
VSYS Max	5.5V

Note that VBUS and VSYS current will depend on use-case, some examples are given in the next section.

Recommended maximum ambient temperature of operation is 70°C.

Chapter 3. Applications information

3.1. Programming the flash

The on-board 2MB QSPI flash can be (re)programmed either using the serial wire debug port or by the special USB mass storage device mode.

The simplest way to reprogram the Pico W's flash is to use the USB mode. To do this, power-down the board, then hold the BOOTSEL button down during board power-up (e.g. hold BOOTSEL down while connecting the USB). The Pico W will then appear as a USB mass storage device. Dragging a special '.uf2' file onto the disk will write this file to the flash and restart the Pico W.

The USB boot code is stored in ROM on RP2040, so can not be accidentally overwritten.

To get started using the SWD port see the Debugging with SWD section in the **Getting started with Raspberry Pi Pico** book.

3.2. General purpose I/O

The Pico W's GPIO is powered from the on-board 3.3V rail, and is fixed at 3.3V.

Pico W exposes 26 of the 30 possible RP2040 GPIO pins by routing them straight out to Pico W header pins. GPIO0 to GPIO22 are digital only, and GPIO 26-28 can be used either as digital GPIO or as ADC inputs (software selectable).

GPIO 26-29 are ADC-capable and have an internal reverse diode to the VDDIO (3.3V) rail, so the input voltage must not exceed VDDIO plus about 300mV. If the RP2040 is unpowered, applying a voltage to these GPIO pins will 'leak' through the diode into the VDDIO rail. GPIO pins 0-25 (and the debug pins) do not have this restriction and therefore voltage can safely be applied to these pins when RP2040 is unpowered up to 3.3V.

3.3. Using the ADC

The RP2040 ADC does not have an on-chip reference; it uses its own power supply as a reference. On Pico W the ADC_AVDD pin (the ADC supply) is generated from the SMPS 3.3V by using an R-C filter (201Ω into 2.2µF).

1. This solution relies on the 3.3V SMPS output accuracy
2. Some PSU noise will not be filtered
3. The ADC draws current (about 150µA if the temperature sense diode is disabled, which can vary between chips); there will be an inherent offset of about 150µA*200 = ~30mV. There is a small difference in current draw when the ADC is sampling (about +20µA), so that offset will also vary with sampling as well as operating temperature.

Changing the resistance between the ADC_VREF and 3.3V pin can reduce the offset at the expense of more noise, which is helpful if the use case can support averaging over multiple samples.

Driving the SMPS mode pin (WL_GPIO1) high forces the power supply into PWM mode. This can greatly reduce the inherent ripple of the SMPS at light load, and therefore reduces the ripple on the ADC supply. This does reduce the power efficiency of the Pico W at light load, so at the end of an ADC conversion PFM mode can be re-enabled by driving WL_GPIO1 low once more. See Section 3.4.

The ADC offset can be reduced by tying a second channel of the ADC to ground, and using this zero measurement as an approximation to the offset.

For much improved ADC performance, an external 3.0V shunt reference, such as LM4040, can be connected from the

ADC_VREF pin to ground. Note that if doing this the ADC range is limited to 0V - 3.0V signals (rather than 0V - 3.3V), and the shunt reference will draw continuous current through the 200Ω filter resistor (3.3V - 3.0V)/200 = ~1.5mA.

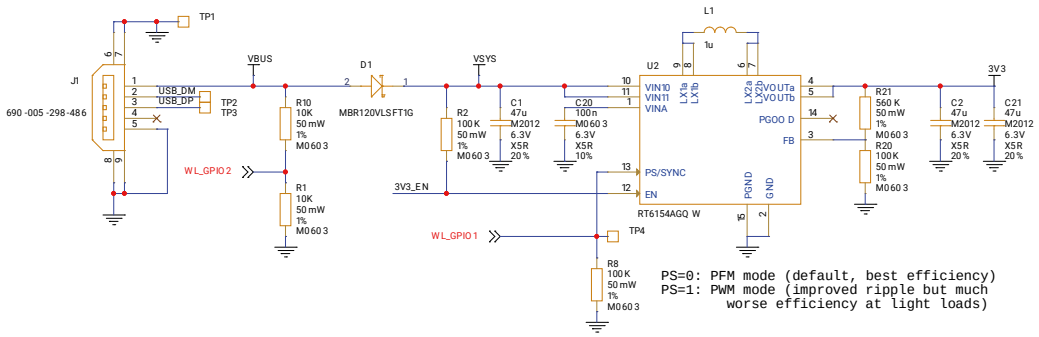
Note that the 1Ω resistor on Pico W (R9) is designed to help with shunt references that would otherwise become unstable when directly connected to 2.2µF. It also ensures there is filtering even in the case that 3.3V and ADC_VREF are shorted together (which users who are tolerant to noise and want to reduce the inherent offset may wish to do).

R7 is a physically large 1608 metric (0603) package resistor, so can be removed easily if a user wants to isolate ADC_VREF and make their own changes to the ADC voltage, for example powering it from an entirely separate voltage (e.g. 2.5V). Note that the ADC on RP2040 has only been qualified at 3.0/3.3V, but should work down to about 2V.

3.4. Powerchain

Pico W has been designed with a simple yet flexible power supply architecture and can easily be powered from other sources such as batteries or external supplies. Integrating the Pico W with external charging circuits is also straightforward. Figure 8 shows the power supply circuitry.

Figure 8. The powerchain of the Pico W Rev3 board.



VBUS is the 5V input from the micro-USB port, which is fed through a Schottky diode to generate VSYS. The VBUS to VSYS diode (D1) adds flexibility by allowing power ORing of different supplies into VSYS.

VSYS is the main system 'input voltage' and feeds the RT6154 buck-boost SMPS, which generates a fixed 3.3V output for the RP2040 device and its I/O (and can be used to power external circuitry). VSYS divided by 3 (by R5, R6 in the Pico W schematic) and can be monitored on ADC channel 3 when a wireless transmission isn't in progress. This can be used for example as a crude battery voltage monitor.

The buck-boost SMPS, as its name implies, can seamlessly switch from buck to boost mode, and therefore can maintain an output voltage of 3.3V from a wide range of input voltages, ~1.8V to 5.5V, which allows a lot of flexibility in the choice of power source.

WL_GPIO2 monitors the existence of VBUS, while R10 and R1 act to pull VBUS down to make sure it is 0V if VBUS is not present.

WL_GPIO1 controls the RT6154 PS (power save) pin. When PS is low (the default on Pico W) the regulator is in pulse frequency modulation (PFM) mode, which, at light loads, saves considerable power by only turning on the switching MOSFETs occasionally to keep the output capacitor topped up. Setting PS high forces the regulator into pulse width modulation (PWM) mode. PWM mode forces the SMPS to switch continuously, which reduces the output ripple considerably at light loads (which can be good for some use cases) but at the expense of much worse efficiency. Note that under heavy load the SMPS will be in PWM mode irrespective of the PS pin state.

The SMPS EN pin is pulled up to VSYS by a 100kΩ resistor and made available on Pico W pin 37. Shorting this pin to ground will disable the SMPS and put it into a low power state.

NOTE

The RP2040 has an on-chip linear regulator (LDO) that powers the digital core at 1.1V (nominal) from the 3.3V supply, which is not shown in Figure 8.

3.5. Powering Raspberry Pi Pico W

The simplest way to power Pico W is to plug in the micro-USB, which will power VSYS (and therefore the system) from the 5V USB VBUS voltage, via D1 (so VSYS becomes VBUS minus the Schottky diode drop).

If the USB port is the **only** power source, VSYS and VBUS can be safely shorted together to eliminate the Schottky diode drop (which improves efficiency and reduces ripple on VSYS).

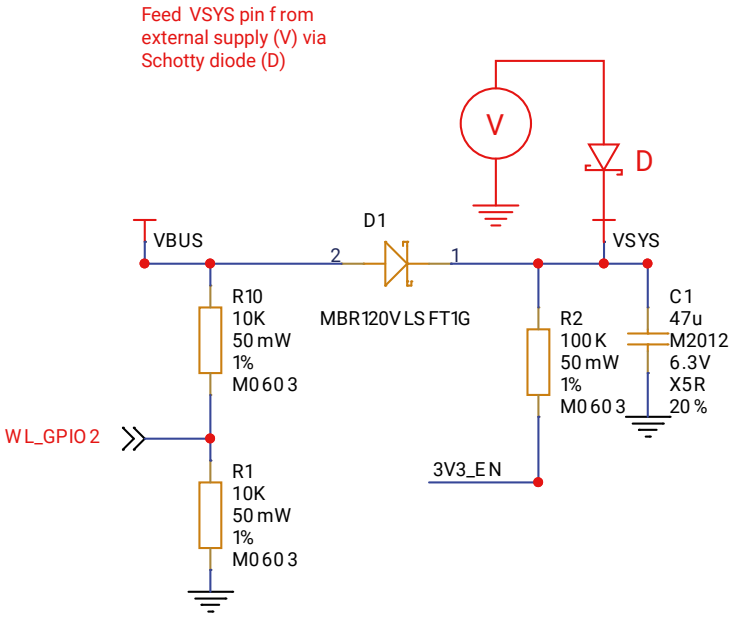
If the USB port is **not** going to be used, it is safe to power Pico W by connecting VSYS to your preferred power source (in the range ~1.8V to 5.5V).

IMPORTANT

If you are using Pico W in USB host mode (e.g. using one of the TinyUSB host examples) then you must power Pico W by providing 5V to the VBUS pin.

The simplest way to safely add a second power source to Pico W is to feed it into VSYS via another Schottky diode (see Figure 9). This will 'OR' the two voltages, allowing the higher of either the external voltage or VBUS to power VSYS, with the diodes preventing either supply from back-powering the other. For example a single Lithium-Ion cell* (cell voltage ~3.0V to 4.2V) will work well, as will three AA series cells (~3.0V to ~4.8V) and any other fixed supply in the range ~2.3V to 5.5V. The downside of this approach is that the second power supply will suffer a diode drop in the same way as VBUS does, and this may not be desirable from an efficiency perspective or if the source is already close to the lower range of input voltage allowed for the RT6154.

Figure 9. Pico W power ORing using diodes.



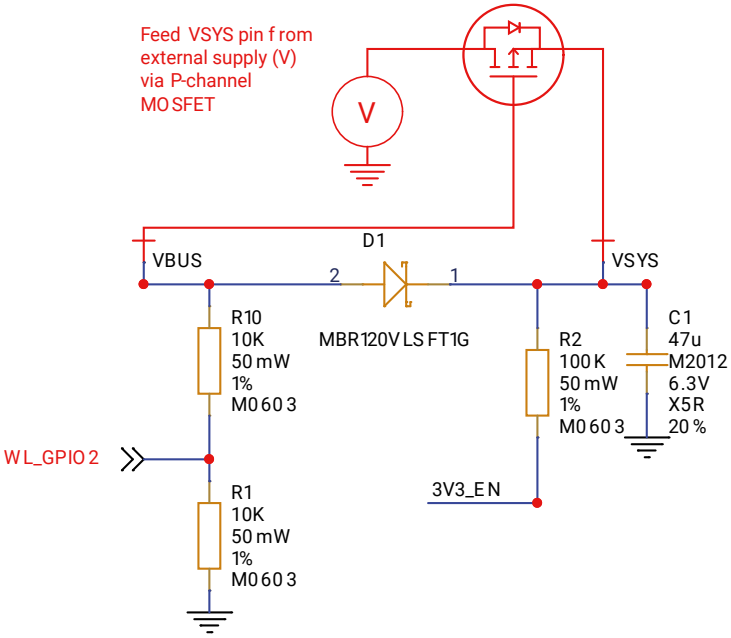
An improved way to power from a second source is using a P-channel MOSFET (P-FET) to replace the Schottky diode as shown in Figure 10. Here, the gate of the FET is controlled by VBUS, and will disconnect the secondary source when VBUS is present. The P-FET should be chosen to have low on resistance, and therefore overcomes the efficiency and

voltage-drop issues with the diode-only solution.

Note that the V_t (threshold voltage) of the P-FET must be chosen to be well below the minimum external input voltage, to make sure the P-FET is turned on swiftly and with low resistance. When the input VBUS is removed, the P-FET will not start to turn on until VBUS drops below the P-FET's V_t, meanwhile the body diode of the P-FET may start to conduct (depending on whether V_t is smaller than the diode drop). For inputs that have a low minimum input voltage, or if the P-FET gate is expected to change slowly (e.g. if any capacitance is added to VBUS) a secondary Schottky diode across the P-FET (in the same direction as the body diode) is recommended. This will reduce the voltage drop across the P-FET's body diode.

An example of a suitable P-MOSFET for most situations is Diodes DMG2305UX which has a maximum V_t of 0.9V and R_{on} of 100mΩ (at 2.5V V_{gs}).

Figure 10. Pico W power ORing using P channel MOSFET.



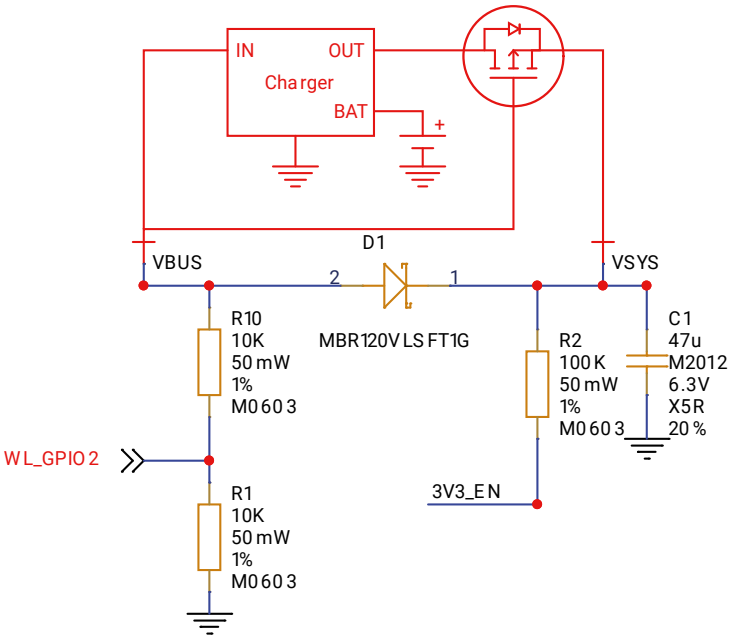
CAUTION

If using Lithium-Ion cells they must have, or be provided with, adequate protection against over-discharge, over-charge, charging outside allowed temperature range, and overcurrent. Bare, unprotected cells are dangerous and can catch fire or explode if over-discharged, over-charged or charged / discharged outside their allowed temperature and/or current range.

3.6. Using a battery charger

Pico W can also be used with a battery charger. Although this is a slightly more complex use case it is still straightforward. Figure 11 shows an example of using a 'power path' type charger (where the charger seamlessly manages swapping between powering from battery or powering from the input source and charging the battery, as needed).

Figure 11. Using Pico W with a charger.



In the example we feed VBUS to the input of the charger, and we feed VSYS with the output via the previously mentioned P-FET arrangement. Depending on your use case you may also want to add a Schottky diode across the P-FET as described in the previous section.

3.7. USB

RP2040 has an integrated USB1.1 PHY and controller which can be used in both device and host mode. Pico W adds the two required 27Ω external resistors and brings this interface to a standard micro-USB port.

The USB port can be used to access the USB bootloader (BOOTSEL mode) stored in the RP2040 boot ROM. It can also be used by user code, to access an external USB device or host.

3.8. Wireless interface

Pico W contains an on-board 2.4GHz wireless interface using the Infineon CYW43439, which has the following features:

- WiFi 4 (802.11n), Single-band (2.4 GHz)
- WPA3
- SoftAP (Up to 4 clients)
- Bluetooth 5.2
    - Support for Bluetooth LE Central and Peripheral roles
    - Support for Bluetooth Classic

The antenna is an onboard antenna licensed from ABRACON (formerly ProAnt). The wireless interface is connected via SPI to the RP2040.

Due to pin limitations, some of the wireless interface pins are shared. The CLK is shared with VSYS monitor, so only when there isn't an SPI transaction in progress can VSYS be read via the ADC. The Infineon CYW43439 DIN/DOUT and IRQ all share one pin on the RP2040. Only when an SPI transaction isn't in progress is it suitable to check for IRQs. The

interface typically runs at 33MHz.

For best wireless performance, the antenna should be in free space. For instance, putting metal under or close by the antenna can reduce its performance both in terms of gain and bandwidth. Adding grounded metal to the sides of the antenna can improve the antenna's bandwidth.

There are three GPIO pins from the CYW43439 that are used for other board functions and can easily be accessed via the SDK:

WL_GPIO2

IP VBUS sense - high if VBUS is present, else low

WL_GPIO1

OP controls the on-board SMPS power save pin (Section 3.4)

WL_GPIO0

OP connected to user LED

NOTE

Full details of the Infineon CYW43439 can be found on the Infineon website.

3.9. Debugging

Pico W brings the RP2040 serial wire debug (SWD) interface to a three-pin debug header. To get started using the debug port see the Debugging with SWD section in the **Getting started with Raspberry Pi Pico** book.

NOTE

The RP2040 chip has internal pull-up resistors on the SWDIO and SWCLK pins, both nominally 60kΩ.

Appendix A: Availability

Raspberry Pi guarantee availability of the Raspberry Pi Pico W product until at least January 2028.

Support

For support see the Pico section of the Raspberry Pi website, and post questions on the Raspberry Pi forum.

Ordering code

Table 1. Part Number

Model	Order Code	EAN	Minimal Order Quantity	RRP
Raspberry Pi Pico W	SC0918	5056561803173	1+ pcs / Bulk	US$6.00
Raspberry Pi Pico WH	SC0919	5056561800196	1+ pcs / Bulk	US$7.00

NOTE

RRP was correct at time of publication and excludes taxes.

Appendix B: Pico W schematic

Figure 12. The Pico W Rev3 board schematic.

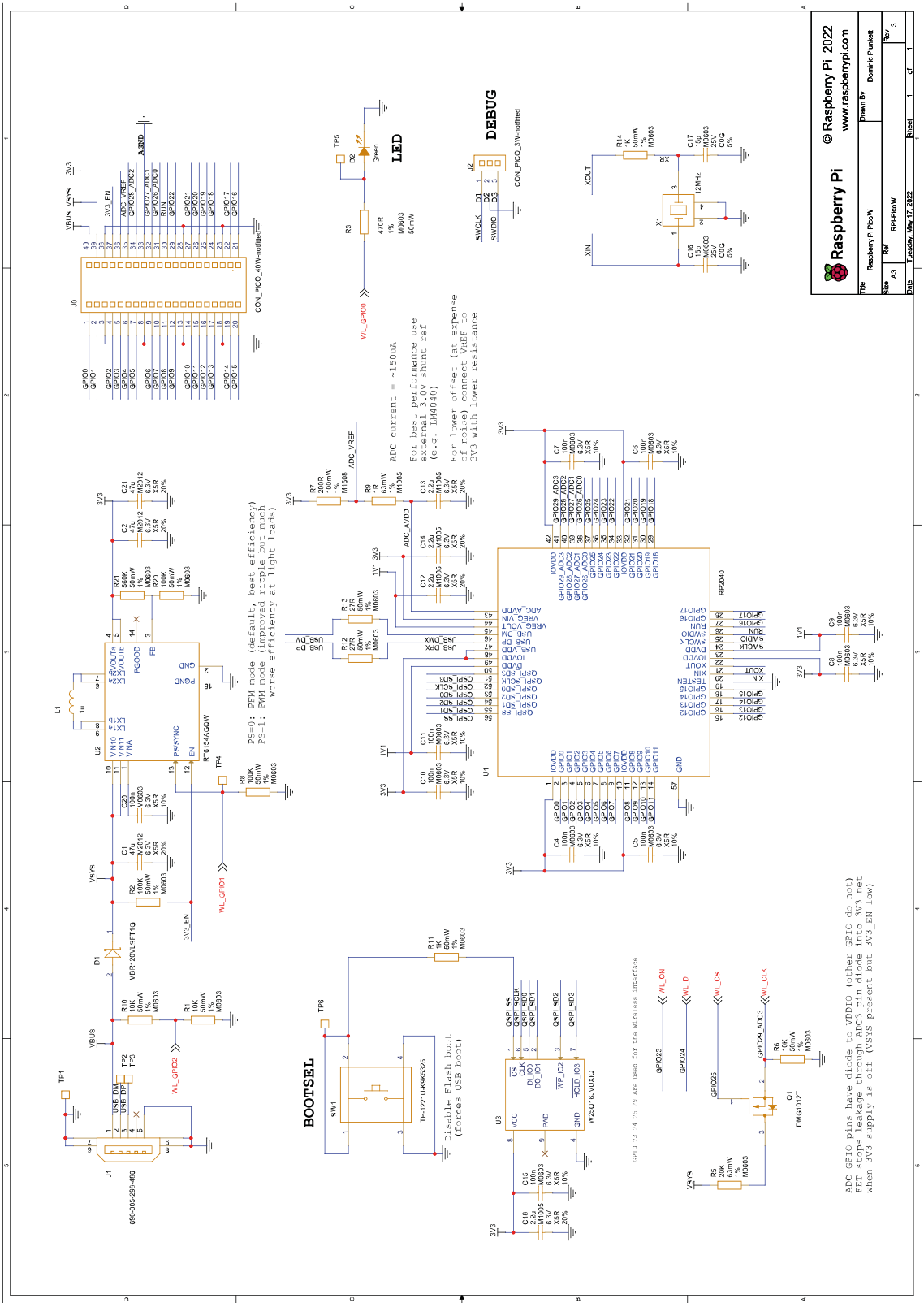
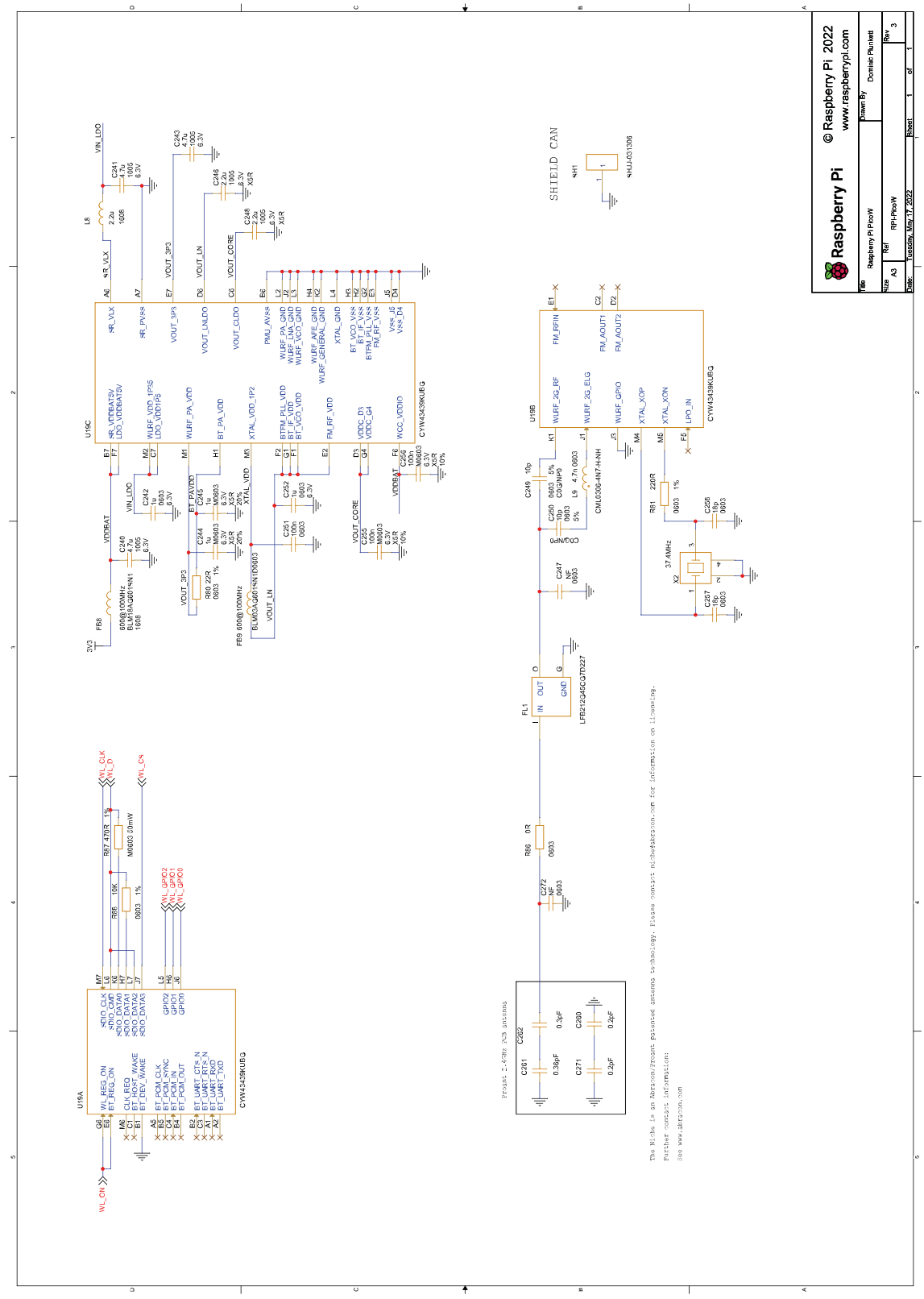
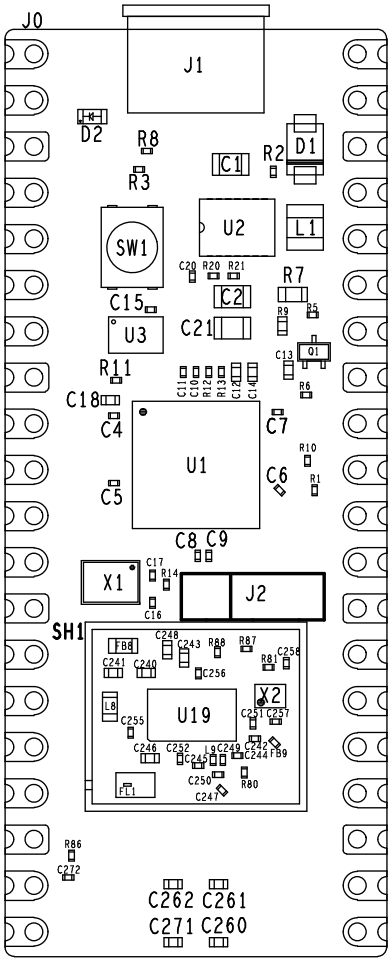


Figure 13. The Pico W Rev3 board schematic.



Appendix C: Pico W component locations

Figure 14. The Pico W Rev3 board component locations.



Appendix D: Documentation release history

Table 2. Documentation release history

Release	Date	Description
1.0	30 Jun 2022	• Initial release
Pico and Pico W databooks combined into a unified release history		
2.0	01 Dec 2022	• Minor updates and corrections • Added RP2040 availability information • Added RP2040 storage conditions and thermal characteristics • Replace SDK library documentation with links to the online version • Updated Picoprobe build and usage instructions
2.1	03 Mar 2023	• A large number of minor updates and corrections • SMT footprint of Pico W corrected • Updated for the 1.5.0 release of the Raspberry Pi Pico C SDK • Added errata E15 • Added documentation around the new Pico Windows Installer • Added documentation around the Pico-W-Go extension for Python development • Added a wireless networking example to the Python documentation • Added package marking specifications • Added RP2040 baseline power consumption figures • Added antenna keep out diagram to Pico W datasheet
2.2	14 Jun 2023	• Minor updates and corrections • Updated for the 1.5.1 release of the Raspberry Pi Pico C SDK • Documentation around Bluetooth support for Pico W

Release	Date	Description
2.3	02 Feb 2024	• Numerous minor updates and corrections • Update ROSC register information • Updated getting started documentation for MS Windows and Apple macOS • Updates arising from the release of Raspberry Pi 5 • Reintroduced updated SDK library documentation (was withdrawn in 2.0 due to XML conflicts) • Updated to include the new recommended part number for crystals used with RP2040 • Added new paste stencil information for Pico and Pico W • Other updates to supporting documentation

The latest release can be found at https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf.

B.2 Datasheet: PH4502C Analog pH Sensor

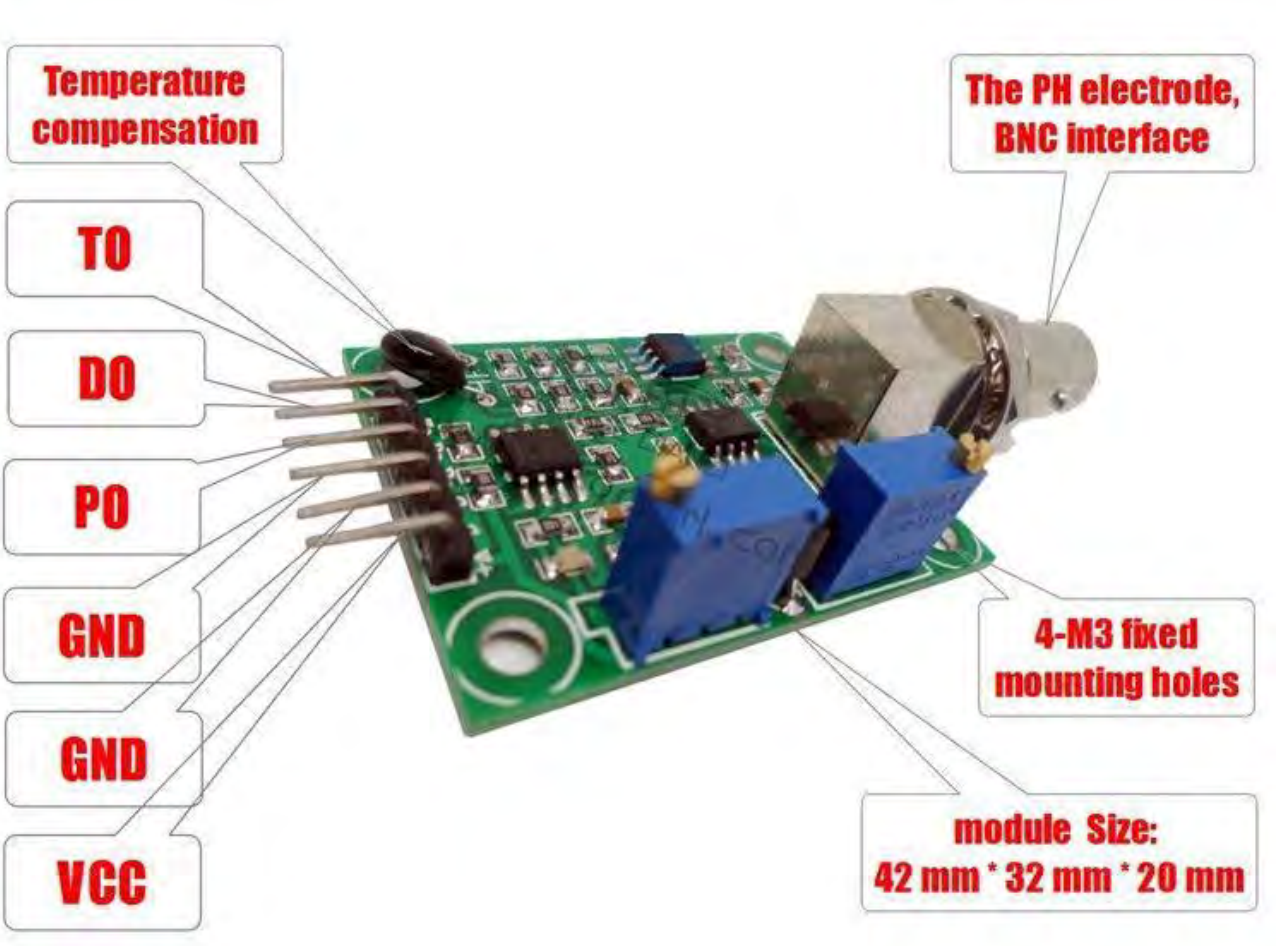
How to use a PH probe and sensor

If you worked with PH metering before you will know that PH values range from 0-14. Where PH 0 Will be very acidic, PH 7 will be neutral and PH 14 very alkaline. Water is near a PH 7 and this is usually around here that we will need to monitor PH of many things. A swimming pool, for example, should be slightly alkaline at 7.2, hydroponics systems around 6 (for optimum plant nutrition takeup) and aquaponics around 6.8.

I wrote this **PH probe and sensor** "how to" because it is not as straightforward as one would think (but quite easy when you understand the ins and outs) mostly because there is not a lot of information on this on the Internet, surely not detailed information.

We will first look at the ph probe module board and then the PH probe because both the PH probe and sensor have to be set correctly:

- offset setting
- limit setting
- sketch to test the board analogue range
- sketch for PH reading and calibration.
- calibration of PH probe
- PH probe usage
  The ph probe module in this tutorial is available on our site here: PH probe module BNC conector



PH Probe Sensor Pinout

TO − Temperature output
DO − 3.3V Output (from ph limit pot)
PO − PH analog output ==> **Arduino A0**
Gnd − Gnd for PH probe (can come from Arduino GND pin) **==> Arduino GND**
Gnd − Gnd for board (can also come from Arduino GND pin) **==> Arduino GND**
VCC − 5V DC (can come from Arduino 5V pin) **==> Arduino 5V pin**
POT 1 − Analog reading offset (Nearest to BNC connector)
POT 2 − PH limit setting

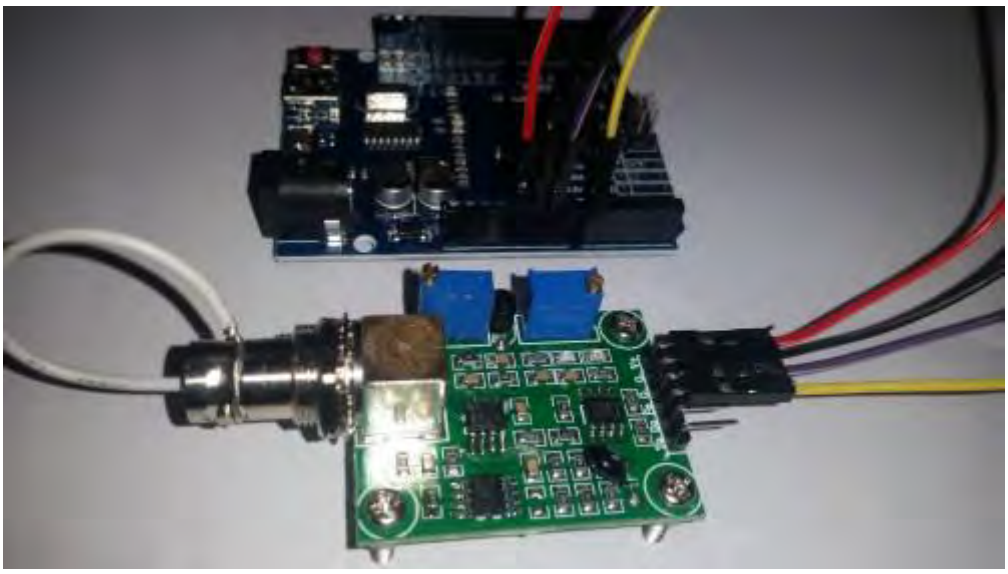
PH probe module Offset and how to use it.

This board have the ability to supply a voltage output to the analogue board that will represent a PH value just like any other sensor that will connect to an analog pin. Ideally, we want a PH 0 represent 0v and a PH of 14 to represent 5V.

BUT there is a catch……, this board by default have PH 7 set to 0V (or near it, it differs from one PH probe to another, that is why we have to calibrate the probe as you will see later on), This means that the voltage will go into the minuses when reading acidic PH values and that cannot be read by the analog Arduino port. The offset pot is used to change this so that a PH 7 will read the expected 2.5V to the Arduino analog pin, the analog pin can read voltages between 0V and 5V hence the 2.5V that is halfway between 0V and 5V as a PH 7 is halfway between PH 0 and PH 14,

You will need to turn the offset potentiometer to get the right offset, The offset pot is the blue pot nearest to the BNC connector.

To set the offset is easy. First, you need to disconnect the probe from the circuit and short-circuit the inside of the BNC connector with the outside to simulate a neutral PH (PH7). I took a piece of wire, strip both sides, wrap the one side around the outside of the BNC connector and push the other side into the BNC hole. This short-circuit represents about a neutral PH reading of 7.



There are two ways you can do the adjustment.

If you have a multimeter handy you can measure the value of the PO pin and adjust the offset potentiometer until PO measures 2.5V.

I prefer to just use the sketch below. Just download it to your Arduino as you will with any other sketch, open serial monitor and view the reading there. All this sketch does is to print the volts it receives from the analog pin and print it to the serial monitor. It of course first changes the digital value to volts to make it easier. Now simply turn the offset pot until it is exactly 2.5V. You can learn more about reading voltages and digital representation of volts here: https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage

Offset sketch

```
void setup() {
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
}


// the loop routine runs over and over showing the voltage on A0
void loop() {
 // read the input on analog pin 0:
 int sensorValue = analogRead(A0);
 // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
 float voltage = sensorValue * (5.0 / 1023.0);
 // print out the value you read:
 Serial.println(voltage);
 delay(300);
}
```

PH limit setting

There is another pot that acts like a limit switch. Basically, the D0 pin on the sensor board will supply 3.3V to the pin until a preset PH value (that you set with the limit pot) is reached, at this point a red LED will light up and the pin will go down to about 0V.

I did not play with this much but suppose it can be handy if you want to activate a buzzer or something if a certain PH is reached, it will work great on an Arduino digital port − that will go high from about 2V up.

This will work if the PH value goes higher than the set value. If you want it to trigger something when the PH goes lower, you need to monitor the digital pin to trigger when the digital pin goes low.

You will unfortunately not be able to set this limit between two values, either if the pH goes up to high or if the PH drop to low. Programmatically you of cause can do an upper and lower limit.

Connecting and calibrating the PH probe.

The hard part is over and this offset does not have to be set again, even if you change PH probes. We have PH probes available here: PH probe Electrode BNC connector
Here is a couple of things to know about PH probes:

1. The probes readings change over time and need to be calibrated every now and again to make sure the value is still the same and be adjusted if it did change.
2. You need at least one PH buffer solution to calibration your PH probe. They are available at many different PH values, A buffer solution of 6.86 and 4.01 is most common as it covers the range of most applications. If you are only going to use one buffer solution make sure its value is near the value range you will use in your normal tests − if it is pool water a buffer solution of 6.86 is usually near enough.
3. Buffers come in pre-made solutions or as a powder. I prefer the powder because it is cheaper and does not have an expiration date. The powder is easy to make up as well, I suppose it depends on the power you will use, the one I use you add the powder to 250ml distilled water and stir until all powder is dissolved. It will last about a month once you added water to it.

A PH probe **takes some time** to get to the right value, allow it to be in the liquid you want to measure for at least two minutes or longer, it does not mean it will be stable at one ph value,

it will jump around a bit between 3 or 4 values but on the last digit, for example, between 6.84 − 6.88

4. PH values differ in different temperatures, although that might sound cumbersome, in the temperature range between 10 − 30 degrees Celcius the PH does not differ and from 30 degrees Celcius it goes up with about a pH of 0.01 to 50 degrees Celcius that is about 0.06. In most uses, it will be below 30 degrees Celcius and temperature do not have to be calculated in.

Hook up your PH probe after you removed the wire you used to short-circuit the BNC connector and download the sketch below.

PH measurement sketch

```
float calibration = 0.00; //change this value to calibrate
const int analogInPin = A0;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
void setup() {
 Serial.begin(9600);
}

void loop() {
 for(int i=0;i<10;i++)
 {
 buf[i]=analogRead(analogInPin);
 delay(30);
 }
 for(int i=0;i<9;i++)
 {
 for(int j=i+1;j<10;j++)
 {
 if(buf[i]>buf[j])
 {
 temp=buf[i];
 buf[i]=buf[j];
 buf[j]=temp;
 }
 }
```

```
 }
 avgValue=0;
 for(int i=2;i<8;i++)
 avgValue+=buf[i];
 float pHVol=(float)avgValue*5.0/1024/6;
 float phValue = -5.70 * pHVol + calibration;
 Serial.print("sensor = ");
 Serial.println(phValue);


 delay(500);
}
```

A note on buffer solutions: do not **CROSS CONTAMINATE!** What I mean by this is to not take the probe from one buffer solution to another or from a liquid sample you tested to a buffer solution without rinsing it thoroughly with distilled water first. You will change your buffer solutions ph and your calibration will be off.

When you place the probe in the first solution you might be surprised at how far off it can be, it's normal though. **Remember to leave the probe in the solution for at least two minutes to stabilise.** In the script's top line you will see a variable called ***"calibration".*** Change this value to the difference between what you see in serial monitor and the buffer solutions value, for example, if you read 5.81 and the buffer solution is 6.86 you should change the variable's value to 2.05.

Upload the changed sketch and see how the value looks now.

Some ideas for you.

You can add a potentiometer to your project and program that to change the calibration for you. You always run a risk that the pot might be adjusted my mistake so a button with a 5-second delay can be programmed to put the unit in calibration mode. This becomes great if you add an LCD screen to it and if you add the calibration value in EEPROM so it holds it even if the PH project is powered off.

How about adding a buzzer to the 3.3V limit output to notify you when the PH is out of range. Usually an upper or lower will be enough, in most applications, you will either be worried about high or low PH values but not both. If this is a problem and you need the PH to be in a certain range you can easily do that programmatically and have the buzzer on a digital pin. If it is crucial you can even add a GSM module to this so you can get an SMS or how about a dosing pump to automatically get the PH back in range.

I just might, but do not hold your breath (so much to do but so little time) :-), do something like that in the future with our cheap Skeleton Duino.

PH probe and sensor conclusion

If you go through these steps once it becomes as easy as pie. All the parts, even those I mention in the ideas section is available on our site.

B.3 Datasheet: IRLZ34NPBF MOSFET

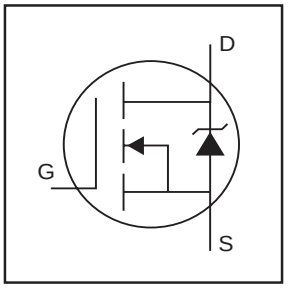
International
IOR Rectifier

PD - 94830

IRLZ34NPbF

HEXFET® Power MOSFET

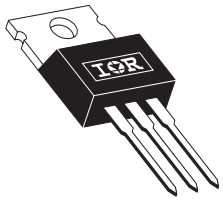
- Logic-Level Gate Drive
- Advanced Process Technology
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- Fully Avalanche Rated
- Lead-Free



V_{DSS} = 55V

$R_{DS(on)}$ = 0.035Ω

I_D = 30A



TO-220AB

Description

Fifth Generation HEXFETs from International Rectifier utilize advanced processing techniques to achieve the lowest possible on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET Power MOSFETs are well known for, provides the designer with an extremely efficient device for use in a wide variety of applications.

The TO-220 package is universally preferred for all commercial-industrial applications at power dissipation levels to approximately 50 watts. The low thermal resistance and low package cost of the TO-220 contribute to its wide acceptance throughout the industry.

Absolute Maximum Ratings

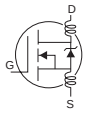
	Parameter	Max.	Units
I_D @ T_C = 25°C	Continuous Drain Current, V_{GS} @ 10V	30	A
I_D @ T_C = 100°C	Continuous Drain Current, V_{GS} @ 10V	21	
I_{DM}	Pulsed Drain Current ①	110	
P_D @T_C = 25°C	Power Dissipation	68	W
	Linear Derating Factor	0.45	W/°C
V_{GS}	Gate-to-Source Voltage	±16	V
E_{AS}	Single Pulse Avalanche Energy ②	110	mJ
I_{AR}	Avalanche Current①	16	A
E_{AR}	Repetitive Avalanche Energy①	6.8	mJ
dv/dt	Peak Diode Recovery dv/dt ③	5.0	V/ns
T_J	Operating Junction and	-55 to + 175	°C
T_{STG}	Storage Temperature Range		
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	
	Mounting torque, 6-32 or M3 screw.	10 lbf•in (1.1N•m)	

Thermal Resistance

	Parameter	Min.	Typ.	Max.	Units
$R_{\theta JC}$	Junction-to-Case	———	———	2.2	°C/W
$R_{\theta CS}$	Case-to-Sink, Flat, Greased Surface	———	0.50	———	
$R_{\theta JA}$	Junction-to-Ambient	———	———	62	

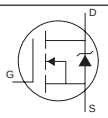
Electrical Characteristics @ T_J = 25°C (unless otherwise specified)

	Parameter	Min.	Typ.	Max.	Units	Conditions
$V_{(BR)DSS}$	Drain-to-Source Breakdown Voltage	55	–––	–––	V	V_{GS} = 0V, I_D = 250µA
$\Delta V_{(BR)DSS}/\Delta T_J$	Breakdown Voltage Temp. Coefficient	–––	0.065	–––	V/°C	Reference to 25°C, I_D = 1mA
$R_{DS(on)}$	Static Drain-to-Source On-Resistance	–––	–––	0.035	Ω	V_{GS} = 10V, I_D = 16A ④
		–––	–––	0.046		V_{GS} = 5.0V, I_D = 16A ④
		–––	–––	0.060		V_{GS} = 4.0V, I_D = 14A ④
$V_{GS(th)}$	Gate Threshold Voltage	1.0	–––	2.0	V	V_{DS} = V_{GS}, I_D = 250µA
g_{fs}	Forward Transconductance	11	–––	–––	S	V_{DS} = 25V, I_D = 16A
I_{DSS}	Drain-to-Source Leakage Current	–––	–––	25	µA	V_{DS} = 55V, V_{GS} = 0V
		–––	–––	250		V_{DS} = 44V, V_{GS} = 0V, T_J = 150°C
I_{GSS}	Gate-to-Source Forward Leakage	–––	–––	100	nA	V_{GS} = 16V
	Gate-to-Source Reverse Leakage	–––	–––	-100		V_{GS} = -16V
Q_g	Total Gate Charge	–––	–––	25	nC	I_D = 16A
Q_{gs}	Gate-to-Source Charge	–––	–––	5.2		V_{DS} = 44V
Q_{gd}	Gate-to-Drain ("Miller") Charge	–––	–––	14		V_{GS} = 5.0V, See Fig. 6 and 13 ④
$t_{d(on)}$	Turn-On Delay Time	–––	8.9	–––	ns	V_{DD} = 28V
t_r	Rise Time	–––	100	–––		I_D = 16A
$t_{d(off)}$	Turn-Off Delay Time	–––	21	–––		R_G = 6.5Ω, V_{GS} = 5.0V
t_f	Fall Time		29			R_D = 1.8Ω, See Fig. 10 ④
L_D	Internal Drain Inductance	–––	4.5	–––	nH	Between lead, 6mm (0.25in.) from package and center of die contact
L_S	Internal Source Inductance	–––	7.5	–––		
C_{iss}	Input Capacitance	–––	880	–––	pF	V_{GS} = 0V
C_{oss}	Output Capacitance	–––	220	–––		V_{DS} = 25V
C_{rss}	Reverse Transfer Capacitance	–––	94	–––		f = 1.0MHz, See Fig. 5



Source-Drain Ratings and Characteristics

	Parameter	Min.	Typ.	Max.	Units	Conditions
I_S	Continuous Source Current (Body Diode)	–––	–––	30	A	MOSFET symbol showing the integral reverse p-n junction diode.
I_{SM}	Pulsed Source Current (Body Diode) ①	–––	–––	110		
V_{SD}	Diode Forward Voltage	–––	–––	1.3	V	T_J = 25°C, I_S = 16A, V_{GS} = 0V ④
t_{rr}	Reverse Recovery Time	–––	76	110	ns	T_J = 25°C, I_F = 16A
Q_{rr}	Reverse RecoveryCharge	–––	190	290	nC	di/dt = 100A/µs ④
t_{on}	Forward Turn-On Time	Intrinsic turn-on time is negligible (turn-on is dominated by L_S+L_D)				



Notes:

① Repetitive rating; pulse width limited by max. junction temperature. (See fig. 11)

② V_{DD} = 25V, starting T_J = 25°C, L = 610µH R_G = 25Ω, I_{AS} = 16A. (See Figure 12)

③ I_{SD} ≤ 16A, di/dt ≤ 270A/µs, V_{DD} ≤ $V_{(BR)DSS}$, T_J ≤ 175°C

④ Pulse width ≤ 300µs; duty cycle ≤ 2%.

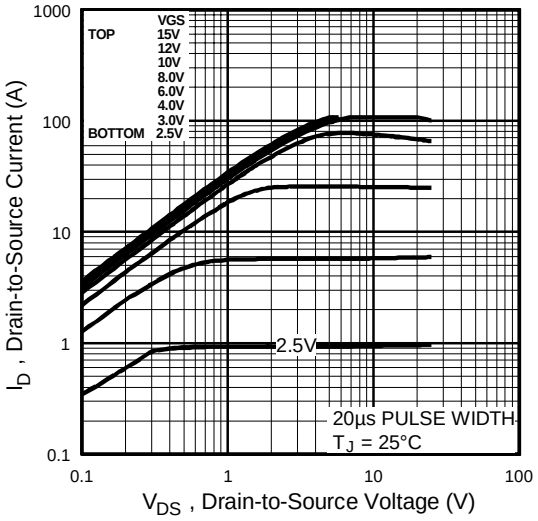


Fig 1. Typical Output Characteristics

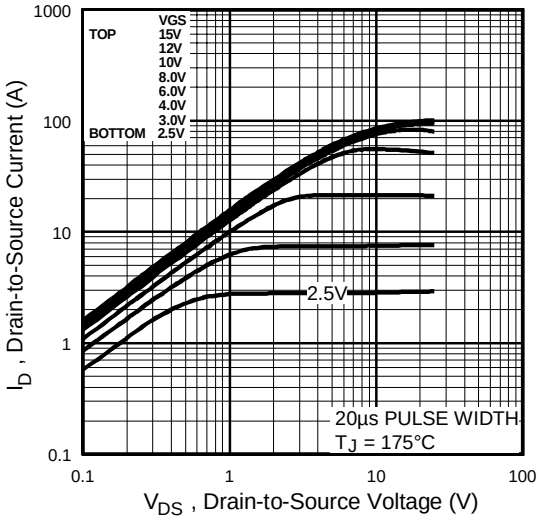


Fig 2. Typical Output Characteristics

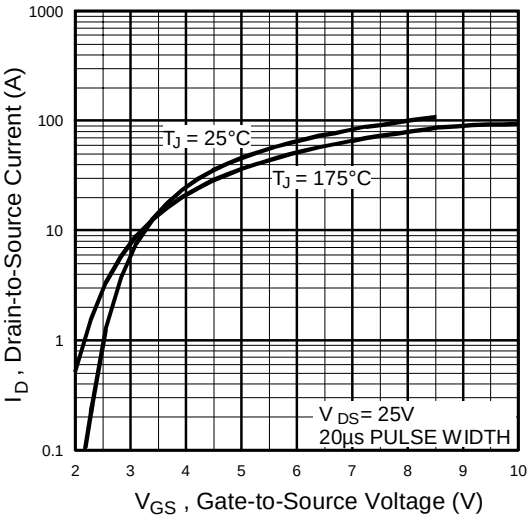


Fig 3. Typical Transfer Characteristics

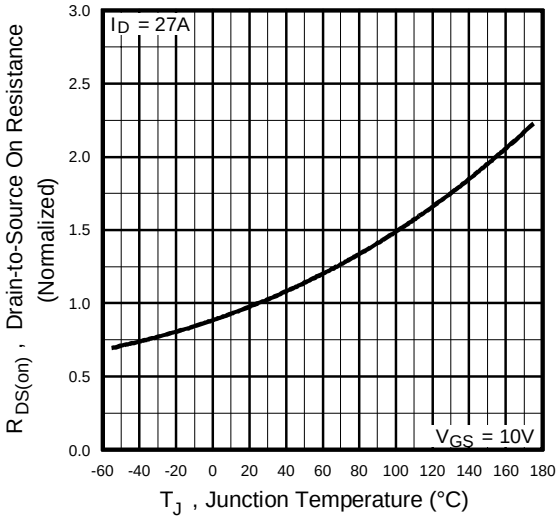


Fig 4. Normalized On-Resistance
Vs. Temperature

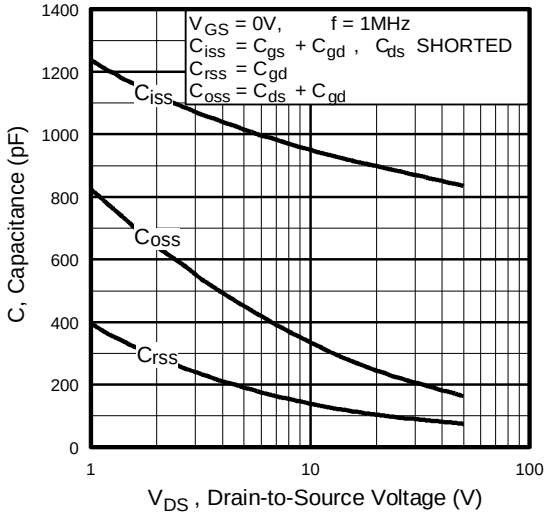


Fig 5. Typical Capacitance Vs. Drain-to-Source Voltage

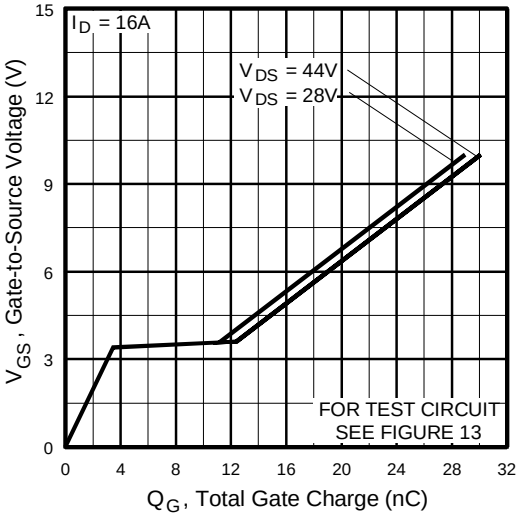


Fig 6. Typical Gate Charge Vs. Gate-to-Source Voltage

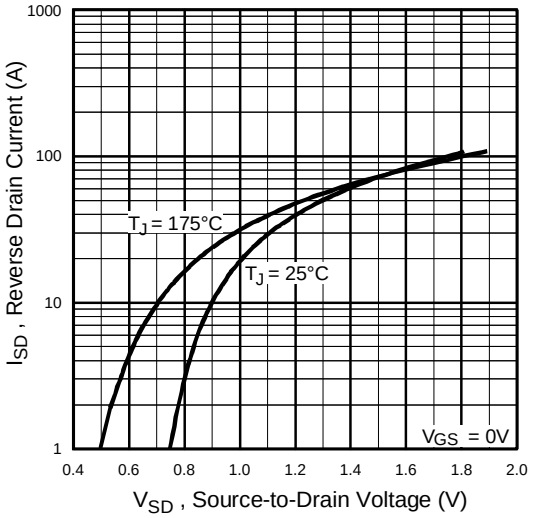


Fig 7. Typical Source-Drain Diode Forward Voltage

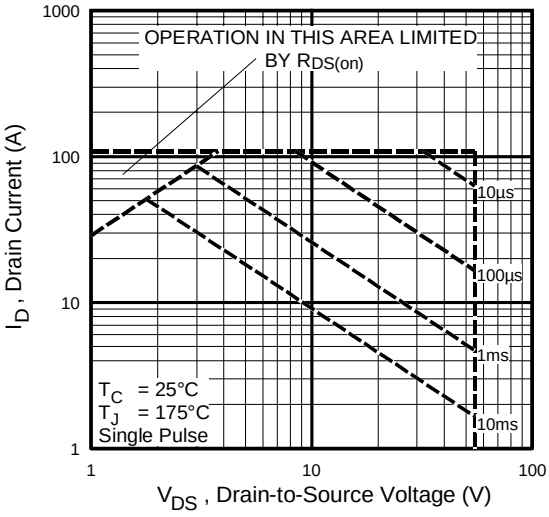


Fig 8. Maximum Safe Operating Area

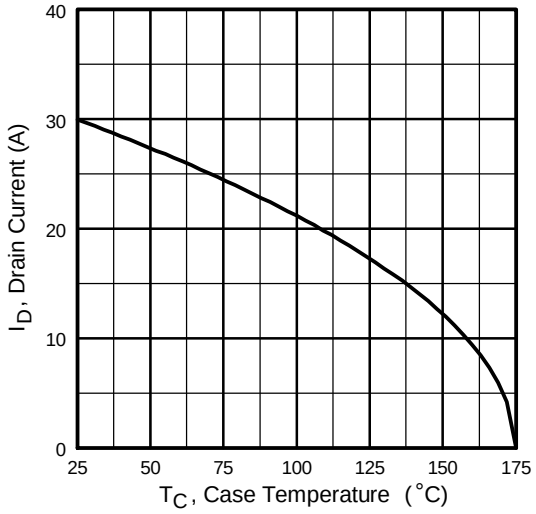


Fig 9. Maximum Drain Current Vs. Case Temperature

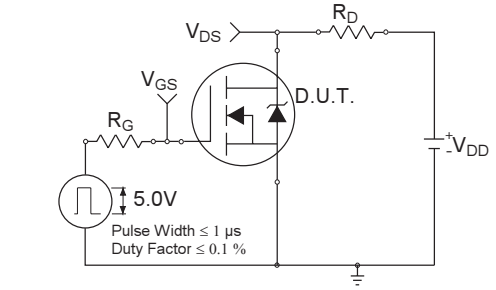


Fig 10a. Switching Time Test Circuit

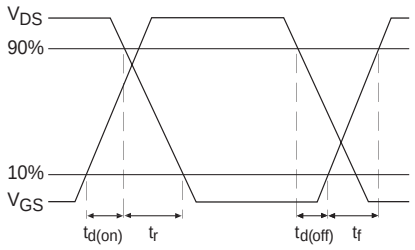


Fig 10b. Switching Time Waveforms

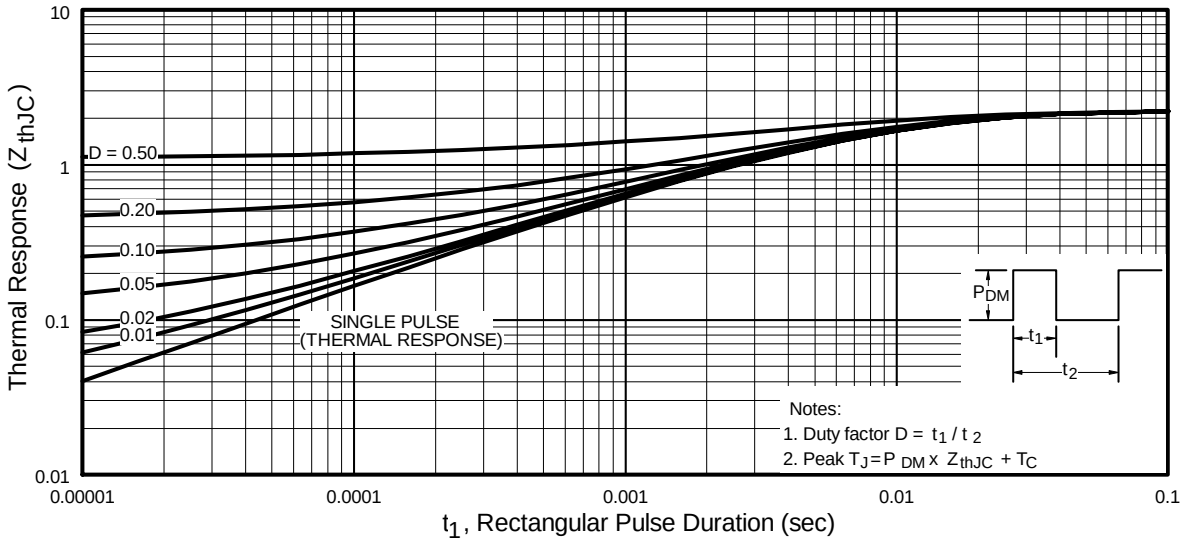


Fig 11. Maximum Effective Transient Thermal Impedance, Junction-to-Case

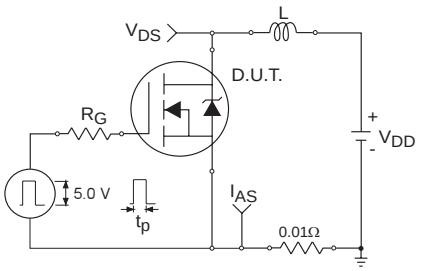


Fig 12a. Unclamped Inductive Test Circuit

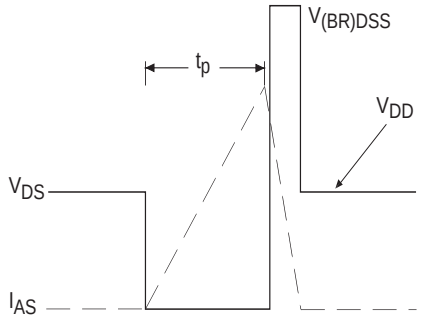


Fig 12b. Unclamped Inductive Waveforms

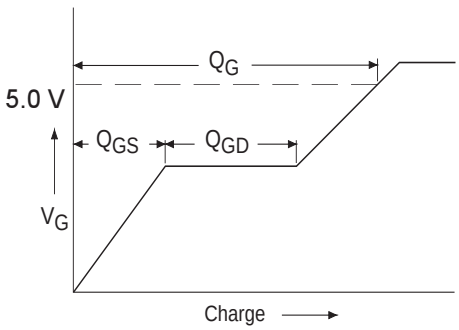


Fig 13a. Basic Gate Charge Waveform

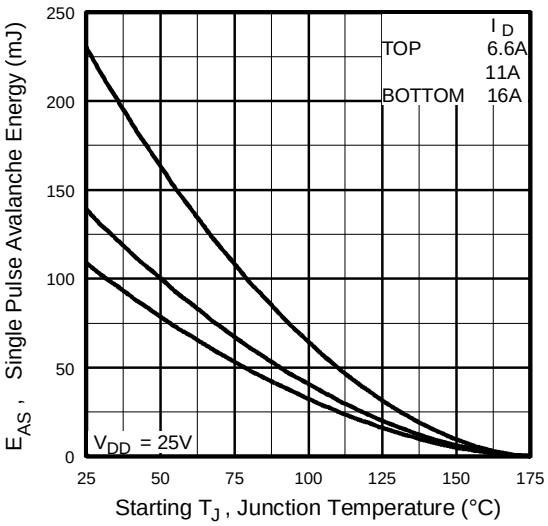


Fig 12c. Maximum Avalanche Energy Vs. Drain Current

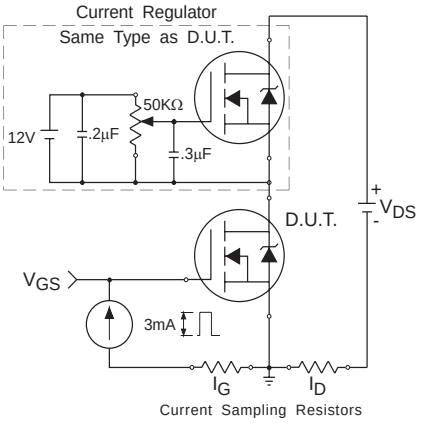
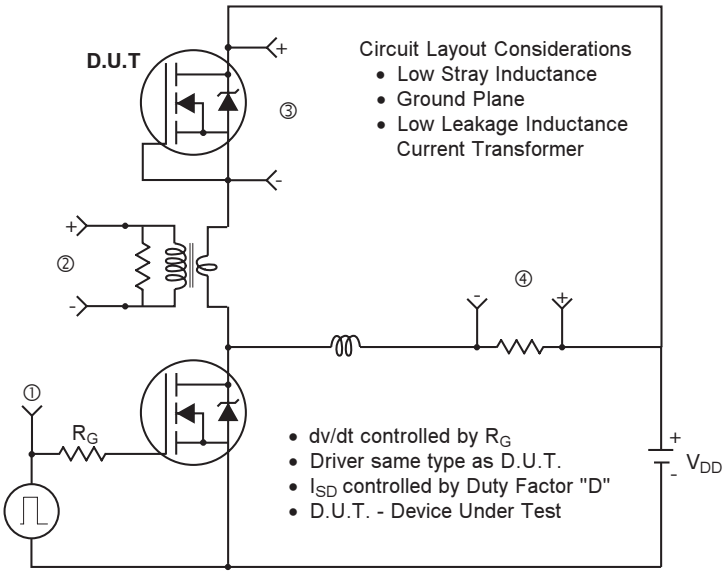
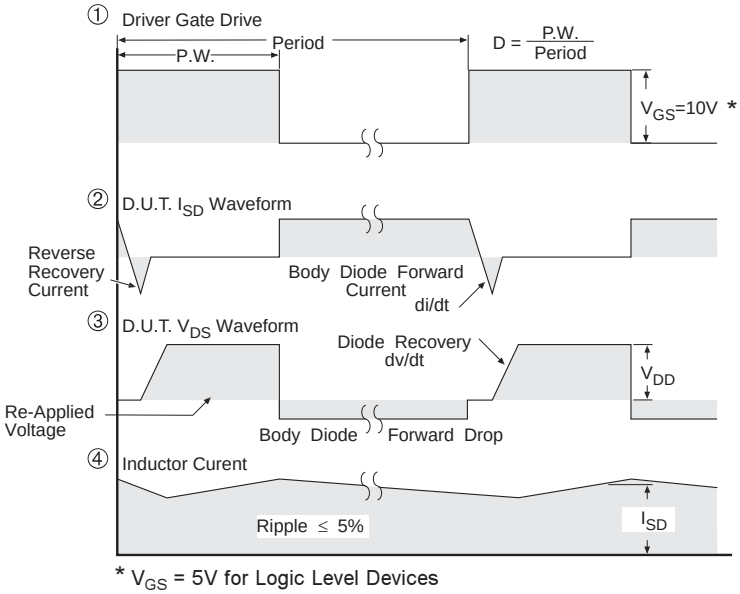


Fig 13b. Gate Charge Test Circuit

Peak Diode Recovery dv/dt Test Circuit



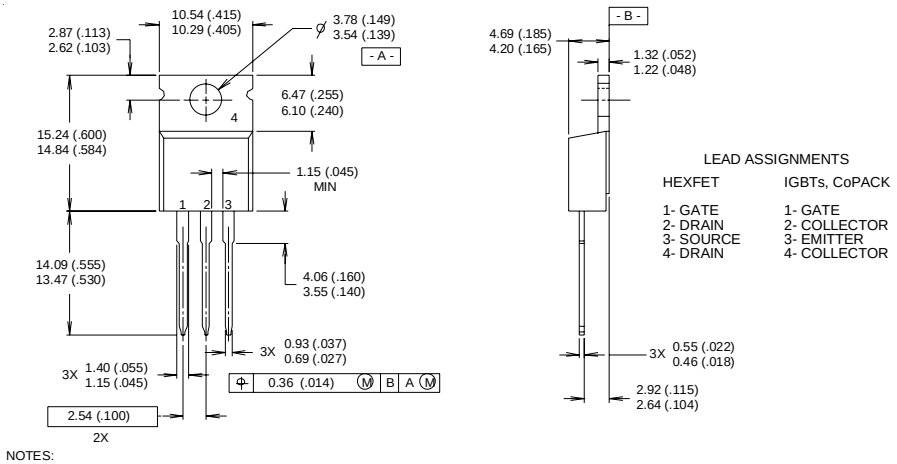


* V_{GS} = 5V for Logic Level Devices

Fig 14. For N-Channel HEXFETS

TO-220AB Package Outline

Dimensions are shown in millimeters (inches)



LEAD ASSIGNMENTS

HEXFET	IGBTs, CoPACK
1- GATE	1- GATE
2- DRAIN	2- COLLECTOR
3- SOURCE	3- EMITTER
4- DRAIN	4- COLLECTOR

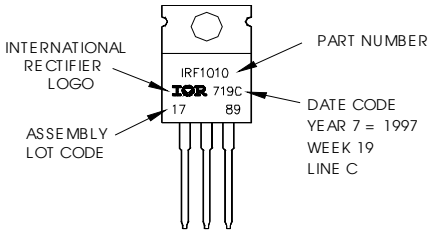
NOTES:

1 DIMENSIONING & TOLERANCING PER ANSI Y14.5M, 1982.
2 CONTROLLING DIMENSION : INCH
3 OUTLINE CONFORMS TO JEDEC OUTLINE TO-220AB.
4 HEATSINK & LEAD MEASUREMENTS DO NOT INCLUDE BURRS.

TO-220AB Part Marking Information

EXAMPLE: THIS IS AN IRF1010
LOT CODE 1789
ASSEMBLED ON WW 19, 1997
IN THE ASSEMBLY LINE "C"

Note: "P" in assembly line position indicates "Lead-Free"



Data and specifications subject to change without notice.

International
IOR Rectifier

IR WORLD HEADQUARTERS: 233 Kansas St., El Segundo, California 90245, USA Tel: (310) 252-7105
TAC Fax: (310) 252-7903
Visit us at www.irf.com for sales contact information.11/03

Note: For the most current drawings please refer to the IR website at:
http://www.irf.com/package/

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie") .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.

B.4 Datasheet: 1N4001RLG Diode

Axial-Lead Glass Passivated Standard Recovery Rectifiers

1N4001, 1N4002, 1N4003, 1N4004, 1N4005, 1N4006, 1N4007

This data sheet provides information on subminiature size, axial lead mounted rectifiers for general−purpose low−power applications.

Features

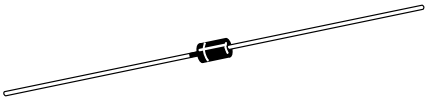
- Shipped in Plastic Bags, 1000 per bag
- Available Tape and Reeled, 5000 per reel, by adding a “RL” suffix to the part number
- Available in Fan−Fold Packaging, 3000 per box, by adding a “FF” suffix to the part number
- Pb−Free Packages are Available

Mechanical Characteristics

- Case: Epoxy, Molded
- Weight: 0.4 gram (approximately)
- Finish: All External Surfaces Corrosion Resistant and Terminal Leads are Readily Solderable
- Lead and Mounting Surface Temperature for Soldering Purposes: 260°C Max. for 10 Seconds, 1/16 in. from case
- Polarity: Cathode Indicated by Polarity Band

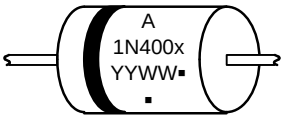
*For additional information on our Pb−Free strategy and soldering details, please download the **onsemi** Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

LEAD MOUNTED RECTIFIERS
50−1000 VOLTS
DIFFUSED JUNCTION



CASE 59−10
AXIAL LEAD
PLASTIC

MARKING DIAGRAM



A = Assembly Location
1N400x = Device Number
x = 1, 2, 3, 4, 5, 6 or 7
YY = Year
WW = Work Week
▪ = Pb−Free Package
(Note: Microdot may be in either location)

ORDERING INFORMATION

See detailed ordering and shipping information on page 5 of this data sheet.

MAXIMUM RATINGS

Rating	Symbol	1N4001	1N4002	1N4003	1N4004	1N4005	1N4006	1N4007	Unit
†Peak Repetitive Reverse Voltage Working Peak Reverse Voltage DC Blocking Voltage	V_{RRM} V_{RWM} V_R	50	100	200	400	600	800	1000	V
†Non−Repetitive Peak Reverse Voltage (halfwave, single phase, 60 Hz)	V_{RSM}	60	120	240	480	720	1000	1200	V
†RMS Reverse Voltage	$V_{R(RMS)}$	35	70	140	280	420	560	700	V
†Average Rectified Forward Current (single phase, resistive load, 60 Hz, T_A = 75°C)	I_O	1.0							A
†Non−Repetitive Peak Surge Current (surge applied at rated load conditions)	I_{FSM}	30 (for 1 cycle)							A
Operating and Storage Junction Temperature Range	T_J T_{stg}	−65 to +150							°C

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.
†Indicates JEDEC Registered Data

THERMAL CHARACTERISTICS

Rating	Symbol	Max	Unit
Maximum Thermal Resistance, Junction−to−Ambient	$R_{\theta JA}$	Note 1	°C/W

ELECTRICAL CHARACTERISTICS†

Rating	Symbol	Typ	Max	Unit
Maximum Instantaneous Forward Voltage Drop, (i_F = 1.0 Amp, T_J = 25°C)	v_F	0.93	1.1	V
Maximum Full−Cycle Average Forward Voltage Drop, (I_O = 1.0 Amp, T_L = 75°C, 1 inch leads)	$V_{F(AV)}$	−	0.8	V
Maximum Reverse Current (rated DC voltage) (T_J = 25°C) (T_J = 100°C)	I_R	 0.05 1.0	 10 50	μA
Maximum Full−Cycle Average Reverse Current, (I_O = 1.0 Amp, T_L = 75°C, 1 inch leads)	$I_{R(AV)}$	−	30	μA

Product parametric performance is indicated in the Electrical Characteristics for the listed test conditions, unless otherwise noted. Product performance may not be indicated by the Electrical Characteristics if operated under different conditions.
†Indicates JEDEC Registered Data

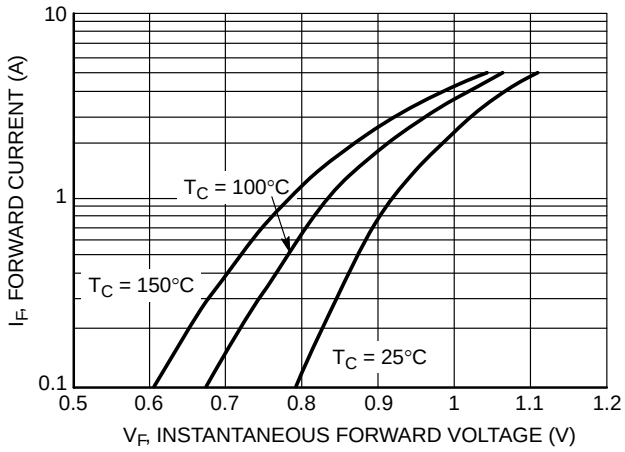


Figure 1. Typical Forward Voltage

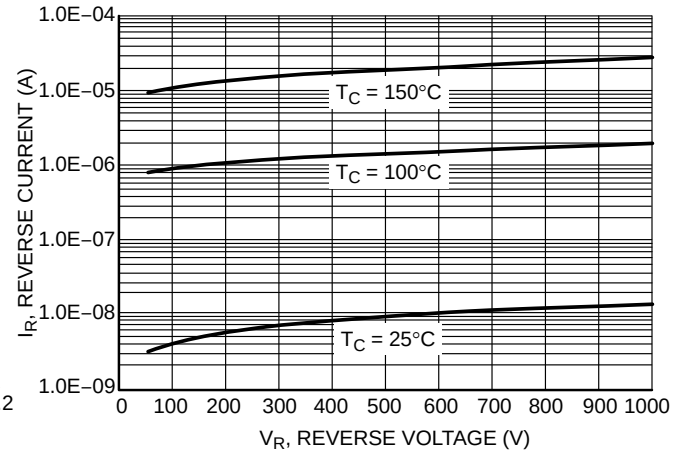


Figure 2. Typical Reverse Current

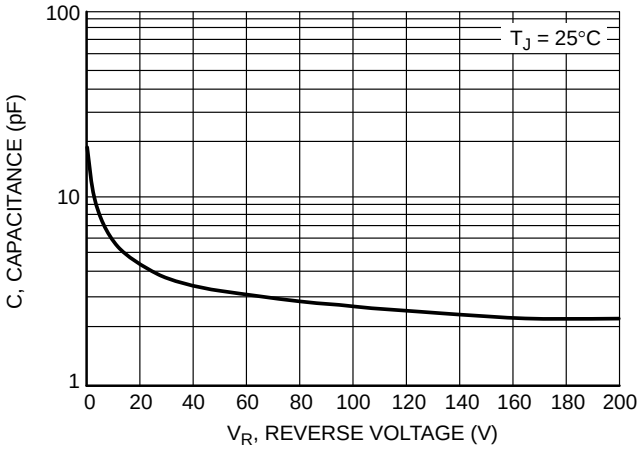


Figure 3. Typical Capacitance

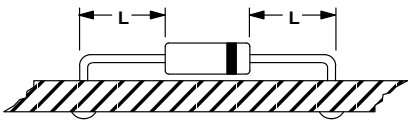
NOTE 1. – AMBIENT MOUNTING DATA

Data shown for thermal resistance, junction−to−ambient ($R_{\theta JA}$) for the mountings shown is to be used as typical guideline values for preliminary engineering or in case the tie point temperature cannot be measured.

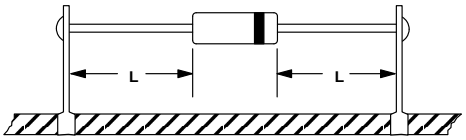
TYPICAL VALUES FOR $R_{\theta JA}$ IN STILL AIR

Mounting Method		Lead Length, L			Units
		1/8	1/4	1/2	
1	$R_{\theta JA}$	52	65	72	°C/W
2		67	80	87	°C/W
3		50			°C/W

MOUNTING METHOD 1

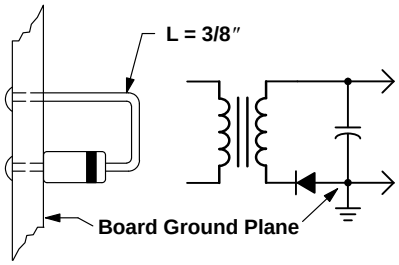


MOUNTING METHOD 2



Vector Pin Mounting

MOUNTING METHOD 3



P.C. Board with
1–1/2″ X 1–1/2″ Copper Surface

ORDERING INFORMATION

Device	Package	Shipping†
1N4001	Axial Lead*	1000 Units/Bag
1N4001G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4001RL	Axial Lead*	5000/Tape & Reel
1N4001RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4002	Axial Lead*	1000 Units/Bag
1N4002G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4002RL	Axial Lead*	5000/Tape & Reel
1N4002RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4003	Axial Lead*	1000 Units/Bag
1N4003G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4003RL	Axial Lead*	5000/Tape & Reel
1N4003RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4004	Axial Lead*	1000 Units/Bag
1N4004G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4004RL	Axial Lead*	5000/Tape & Reel
1N4004RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4005	Axial Lead*	1000 Units/Bag
1N4005G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4005RL	Axial Lead*	5000/Tape & Reel
1N4005RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4006	Axial Lead*	1000 Units/Bag
1N4006G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4006FFG	Axial Lead* (Pb−Free)	3000 Units/Box
1N4006RL	Axial Lead*	5000/Tape & Reel
1N4006RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel
1N4007	Axial Lead*	1000 Units/Bag
1N4007G	Axial Lead* (Pb−Free)	1000 Units/Bag
1N4007FFG	Axial Lead* (Pb−Free)	3000 Units/Box
1N4007RL	Axial Lead*	5000/Tape & Reel
1N4007RLG	Axial Lead* (Pb−Free)	5000/Tape & Reel

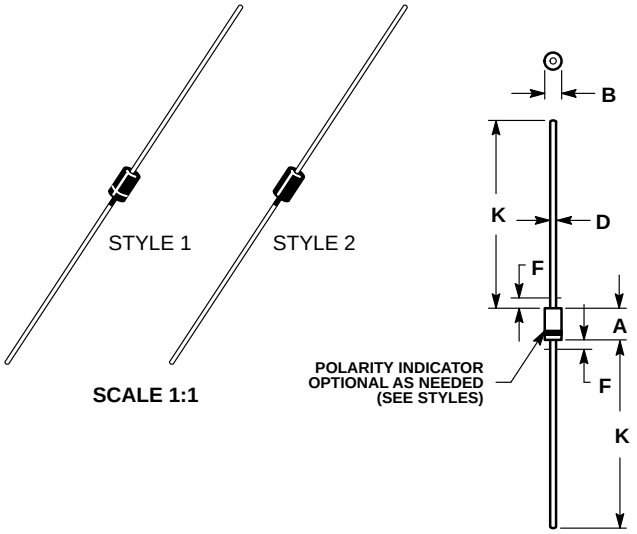
†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specifications Brochure, BRD8011/D.
*This package is inherently Pb−Free.

MECHANICAL CASE OUTLINE
PACKAGE DIMENSIONS

ON Semiconductor®

AXIAL LEAD
CASE 59−10
ISSUE U

DATE 15 FEB 2005



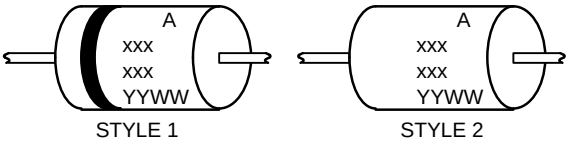
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. ALL RULES AND NOTES ASSOCIATED WITH JEDEC DO−41 OUTLINE SHALL APPLY
4. POLARITY DENOTED BY CATHODE BAND.
5. LEAD DIAMETER NOT CONTROLLED WITHIN F DIMENSION.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.161	0.205	4.10	5.20
B	0.079	0.106	2.00	2.70
D	0.028	0.034	0.71	0.86
F	−−−	0.050	−−−	1.27
K	1.000	−−−	25.40	−−−

STYLE 1:
PIN 1. CATHODE (POLARITY BAND)
2. ANODE

STYLE 2:
NO POLARITY

GENERIC MARKING DIAGRAM*



XXX	= Specific Device Code
A	= Assembly Location
YY	= Year
WW	= Work Week

*This information is generic. Please refer to device data sheet for actual part marking. Pb−Free indicator, “G” or microdot “ ▪”, may or may not be present.

DOCUMENT NUMBER:	**98ASB42045B**	Electronic versions are uncontrolled except when accessed directly from the Document Repository. Printed versions are uncontrolled except when stamped “CONTROLLED COPY” in red.	
DESCRIPTION:	**AXIAL LEAD**		**PAGE 1 OF 1**

ADDITIONAL INFORMATION

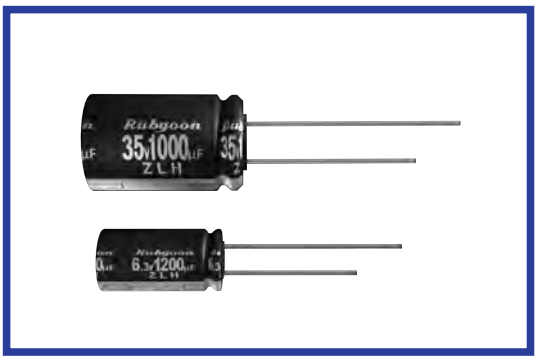
TECHNICAL PUBLICATIONS:
Technical Library: www.onsemi.com/design/resources/technical−documentation
onsemi Website: www.onsemi.com

ONLINE SUPPORT: www.onsemi.com/support
For additional information, please contact your local Sales Representative at www.onsemi.com/support/sales

B.5 Datasheet: 16ZLH220MEFCT16.3X11 Capacitor

ZLH series

105℃ 6000～10000時間品　小形化 長寿命 低インピーダンス品
105℃ 6000～10000 hours, Miniaturized, Long Life, Low Impedance



RoHS compliance

◆規格表／SPECIFICATIONS

項目　Item	特性　Characteristics
カテゴリ温度範囲 Category Temperature Range	-40～+105℃
定格電圧範囲 Rated Voltage Range	6.3～100Vdc
静電容量許容差 Capacitance Tolerance	±20%（20℃, 120Hz）
漏れ電流 Leakage Current (MAX)	I＝0.01CV又は3μAのいずれか大なる値以下（定格電圧印加2分後） I＝0.01CV or 3μA whichever is greater.　（After 2 minutes） I=漏れ電流（μA） C=静電容量（μF） V=定格電圧（Vdc） Leakage Current　Capacitance　Rated Voltage
損失角の正接（tanδ） Dissipation Factor (MAX)	定格電圧（Vdc） Rated Voltage: 6.3 / 10 / 16 / 25 / 35 / 50 / 63 / 80 / 100 tanδ: 0.22 / 0.19 / 0.16 / 0.14 / 0.12 / 0.10 / 0.09 / 0.08 / 0.08　（20℃, 120Hz） 1000μFを越えるものは1000μF増す毎に上表の値に0.02を加えた値とする。 When capacitance is over 1000μF, tanδ shall be added 0.02 to the listed value with increase of every 1000μF.
耐久性 Endurance	105℃中で右表の時間定格電圧（リプル重畳）印加後、下記項目を満足すること。 After applying rated voltage with rated ripple current for specified time at 105℃, the capacitors shall meet the following requirements. 静電容量変化率 Capacitance Change: 初期値の±25%以内（6.3Vdc, 10Vdc：±30%） Within ±25% of the initial value (6.3Vdc, 10Vdc：±30%) 損失角の正接 Dissipation Factor: 規格値の 200% 以下 Not more than 200% of the specified value. 漏れ電流 Leakage Current: 規格値以下 Not more than the specified value. ケースサイズ Case Size / 時間(hrs) Time(hrs): φD≦6.3 / 6000; φD＝8 / 8000; φD≧10 / 10000
低温特性 Low Temperature Stability （インピーダンス比） Impedance Ratio (MAX)	定格電圧（Vdc） Rated Voltage: 6.3 / 10 / 16 / 25 / 35 / 50 / 63 / 80 / 100 Z(-25℃)/Z(+20℃): 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2　（120Hz） Z(-40℃)/Z(+20℃): 3 / 3 / 3 / 3 / 3 / 3 / 3 / 3 / 3

◆呼称方法／PART NUMBER

□□□	ZLH	□□□□□	M	□□□	□□	D x L
定格電圧 Rated Voltage	シリーズ名 Series	静電容量 Capacitance	静電容量許容差 Capacitance Tolerance	副記号 Option	リード加工記号 Lead Forming	ケースサイズ Case Size

◆リプル電流補正係数／MULTIPLIER FOR RIPPLE CURRENT

周波数（Hz） Frequency		120	1k	10k	100k≦
係数 Coefficient	8.2～33μF	0.42	0.70	0.90	1.00
	47～270μF	0.50	0.73	0.92	1.00
	330～680μF	0.55	0.77	0.94	1.00
	820～1800μF	0.60	0.80	0.96	1.00
	2200～8200μF	0.70	0.85	0.98	1.00

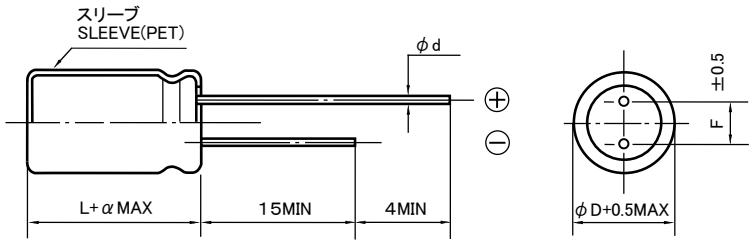
◆副記号／OPTION

EFC：PETスリーブ PET Sleeve

リード線形アルミニウム電解コンデンサ
RADIAL LEAD ALUMINUM ELECTROLYTIC CAPACITORS

ZLH

◆寸法図／DIMENSIONS



(mm)

φD	5	6.3	8	10	12.5	16	18
φd	0.5		0.6			0.8	
F	2.0	2.5	3.5	5.0		7.5	
α	L≦16 : α=1.5 L≧20 : α=2.0						

◆標準品一覧表／STANDARD SIZE

Rated Ripple Current(mA r.m.s./105℃, 100kHz)

定格電圧 Rated Voltage (Vdc)	静電容量 Capacitance (μF)	外形寸法 Size φD×L (mm)	定格リプル電流 RatedRipple Current	インピーダンス (Ω MAX) Impedance	
				20℃, 100kHz	-10℃, 100kHz
6.3	220	5×11	345	0.22	0.80
	470	6.3×11	540	0.094	0.35
	820	8×11.5	945	0.056	0.19
	1200	8×16	1250	0.045	0.15
	1200	10×12.5	1330	0.039	0.14
	1500	8×20	1500	0.029	0.11
	1800	10×16	1760	0.028	0.10
	2200	10×20	1960	0.020	0.060
	2700	10×23	2250	0.018	0.054
	3900	12.5×20	2480	0.017	0.043
	4700	12.5×25	2900	0.015	0.038
	5600	12.5×30	3450	0.013	0.033
	6800	16×20	3250	0.015	0.038
	6800	12.5×35	3570	0.012	0.031
	8200	16×25	3630	0.013	0.035
10	150	5×11	345	0.22	0.80
	330	6.3×11	540	0.094	0.35
	680	8×11.5	945	0.056	0.19
	1000	8×16	1250	0.045	0.15
	1000	10×12.5	1330	0.039	0.14
	1500	8×20	1500	0.029	0.11
	1500	10×16	1760	0.028	0.10
	1800	10×20	1960	0.020	0.060
	2200	10×23	2250	0.018	0.054
	3300	12.5×20	2480	0.017	0.043
	3900	12.5×25	2900	0.015	0.038
	4700	12.5×30	3450	0.013	0.033
	4700	16×20	3250	0.015	0.038
	5600	12.5×35	3570	0.012	0.031
	6800	16×25	3630	0.013	0.035
16	100	5×11	345	0.22	0.80
	220	6.3×11	540	0.094	0.35
	470	8×11.5	945	0.056	0.19
	680	8×16	1250	0.045	0.15
	680	10×12.5	1330	0.039	0.14
	1000	8×20	1500	0.029	0.11
	1000	10×16	1760	0.028	0.10
	1500	10×20	1960	0.020	0.06
	1800	10×23	2250	0.018	0.054
	2200	12.5×20	2480	0.017	0.043
	2700	12.5×25	2900	0.015	0.038
	3300	12.5×30	3450	0.013	0.033
	3300	16×20	3250	0.015	0.038
	3900	12.5×35	3570	0.012	0.031
	4700	16×25	3630	0.013	0.035

定格電圧 Rated Voltage (Vdc)	静電容量 Capacitance (μF)	外形寸法 Size φD×L (mm)	定格リプル電流 RatedRipple Current	インピーダンス (Ω MAX) Impedance	
				20℃, 100kHz	-10℃, 100kHz
25	68	5×11	345	0.220	0.80
	150	6.3×11	540	0.094	0.35
	330	8×11.5	945	0.056	0.19
	390	8×16	1250	0.045	0.15
	470	10×12.5	1330	0.039	0.14
	560	8×20	1500	0.029	0.11
	680	10×16	1760	0.028	0.10
	820	10×20	1960	0.020	0.060
	1000	10×23	2250	0.018	0.054
	1500	12.5×20	2480	0.017	0.043
	1800	12.5×25	2900	0.015	0.038
	2200	12.5×30	3450	0.013	0.033
	2200	16×20	3250	0.015	0.038
	2700	12.5×35	3570	0.012	0.031
	3300	16×25	3630	0.013	0.035
35	47	5×11	345	0.220	0.80
	100	6.3×11	540	0.094	0.35
	220	8×11.5	945	0.056	0.19
	270	8×16	1250	0.045	0.15
	330	10×12.5	1330	0.039	0.14
	390	8×20	1500	0.029	0.11
	470	10×16	1760	0.028	0.10
	560	10×20	1960	0.020	0.060
	680	10×23	2250	0.018	0.054
	1000	12.5×20	2480	0.017	0.043
	1200	12.5×25	2900	0.015	0.038
	1500	12.5×30	3450	0.013	0.033
	1500	16×20	3250	0.015	0.038
	1800	12.5×35	3570	0.012	0.031
	2200	16×25	3630	0.013	0.035
50	27	5×11	238	0.34	1.18
	56	6.3×11	385	0.14	0.50
	100	8×11.5	724	0.074	0.22
	120	8×16	950	0.061	0.18
	150	10×12.5	979	0.061	0.18
	180	8×20	1190	0.046	0.14
	220	10×16	1370	0.042	0.12
	270	10×20	1580	0.030	0.09
	330	10×23	1870	0.028	0.085
	470	12.5×20	2050	0.027	0.068
	560	12.5×25	2410	0.023	0.059
	680	12.5×30	2860	0.021	0.052
	820	12.5×35	2960	0.019	0.051
	820	16×20	2730	0.023	0.059
	1000	16×25	3010	0.021	0.056

◆標準品一覧表／STANDARD SIZE

Rated Ripple Current(mA r.m.s./105℃, 100kHz)

定格電圧 Rated Voltage (Vdc)	静電容量 Capacitance (μF)	外形寸法 Size φD×L (mm)	定格リプル電流 RatedRipple Current	インピーダンス (Ω MAX) Impedance	
				20℃, 100kHz	-10℃, 100kHz
63	18	5×11	173	0.88	3.5
	47	6.3×11	278	0.35	1.4
	82	8×11.5	525	0.22	0.88
	100	8×16	688	0.16	0.64
	120	10×12.5	725	0.15	0.6
	150	8×20	861	0.12	0.48
	180	10×16	998	0.11	0.44
	270	10×20	1200	0.078	0.31
	270	12.5×16	1200	0.082	0.27
	330	10×23	1410	0.069	0.28
	390	12.5×20	1570	0.060	0.19
	470	12.5×25	1990	0.043	0.14
	560	12.5×30	2410	0.035	0.13
	560	16×20	2100	0.043	0.14
	680	12.5×35	2620	0.033	0.11
	820	12.5×40	2940	0.027	0.09
	820	16×25	2730	0.032	0.096
	820	18×20	2500	0.038	0.100
	1200	16×31.5	2990	0.024	0.068
	1200	18×25	2800	0.031	0.084
	1500	16×35.5	3040	0.021	0.057
	1500	18×31.5	3300	0.025	0.068
	1800	16×40	3570	0.019	0.057
	1800	18×35.5	3570	0.020	0.054
	2200	18×40	3670	0.018	0.049
80	12	5×11	163	1.4	5.6
	33	6.3×11	267	0.57	2.3
	56	8×11.5	462	0.36	1.4
	68	8×16	585	0.25	1.0
	82	10×12.5	624	0.23	0.96
	100	8×20	735	0.19	0.76
	120	10×16	780	0.17	0.72
	180	10×20	1040	0.12	0.52
	180	12.5×16	975	0.13	0.43
	220	10×23	1170	0.11	0.47
	270	12.5×20	1430	0.085	0.31
	330	12.5×25	1620	0.060	0.23
	390	12.5×30	1950	0.051	0.21
	390	16×20	1750	0.058	0.21
	470	12.5×35	2140	0.043	0.17
	560	12.5×40	2340	0.036	0.15
	560	16×25	2210	0.044	0.16
	560	18×20	1950	0.054	0.18
	680	16×31.5	2400	0.033	0.12
	820	16×35.5	2600	0.029	0.10
	820	18×25	2270	0.038	0.13
	1000	16×40	2860	0.027	0.09
	1000	18×31.5	2470	0.031	0.11
	1200	18×35.5	2860	0.027	0.084
	1500	18×40	3510	0.026	0.076

定格電圧 Rated Voltage (Vdc)	静電容量 Capacitance (μF)	外形寸法 Size φD×L (mm)	定格リプル電流 RatedRipple Current	インピーダンス (Ω MAX) Impedance	
				20℃, 100kHz	-10℃, 100kHz
100	8.2	5×11	163	1.4	5.6
	18	6.3×11	267	0.57	2.3
	33	8×11.5	462	0.36	1.4
	47	8×16	585	0.25	1.0
	56	10×12.5	624	0.23	0.96
	68	8×20	735	0.19	0.76
	82	10×16	780	0.17	0.72
	100	10×20	1040	0.12	0.52
	100	12.5×16	975	0.13	0.43
	120	10×23	1170	0.11	0.47
	150	12.5×20	1430	0.085	0.31
	220	12.5×25	1620	0.060	0.23
	270	12.5×30	1950	0.051	0.21
	270	16×20	1750	0.058	0.21
	330	12.5×35	2140	0.043	0.17
	390	12.5×40	2340	0.036	0.15
	390	16×25	2210	0.044	0.16
	390	18×20	1950	0.054	0.18
	470	16×31.5	2400	0.033	0.12
	470	18×25	2270	0.038	0.13
	560	16×35.5	2600	0.029	0.10
	560	18×31.5	2470	0.031	0.11
	680	16×40	2860	0.027	0.09
	680	18×35.5	2860	0.027	0.084
	820	18×40	3510	0.026	0.076

B.6 Datasheet: SN74LS151N Multiplexer

SDLS054

SN54150, SN54151A, SN54LS151, SN54S151, SN74150, SN74151A, SN74LS151, SN74S151
DATA SELECTORS/MULTIPLEXERS

DECEMBER 1972 – REVISED MARCH 1988

- '150 Selects One-of-Sixteen Data Sources
- Others Select One-of-Eight Data Sources
- All Perform Parallel-to-Serial Conversion
- All Permit Multiplexing from N Lines to One Line
- Also For Use as Boolean Function Generator
- Input-Clamping Diodes Simplify System Design
- Fully Compatible with Most TTL Circuits

TYPE	TYPICAL AVERAGE PROPAGATION DELAY TIME DATA INPUT TO W OUTPUT	TYPICAL POWER DISSIPATION
'150	13 ns	200 mW
'151A	8 ns	145 mW
'LS151	13 ns	30 mW
'S151	4.5 ns	225 mW

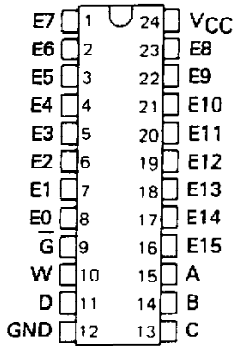
description

These monolithic data selectors/multiplexers contain full on-chip binary decoding to select the desired data source. The '150 selects one-of-sixteen data sources; the '151A, 'LS151, and 'S151 select one-of-eight data sources. The '150, '151A, 'LS151, and 'S151 have a strobe input which must be at a low logic level to enable these devices. A high level at the strobe forces the W output high, and the Y output (as applicable) low.

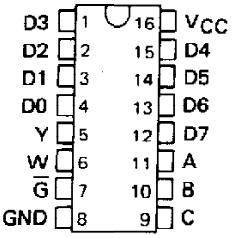
The '150 has only an inverted W output; the '151A, 'LS151, and 'S151 feature complementary W and Y outputs.

The '151A and '152A incorporate address buffers that have symmetrical propagation delay times through the complementary paths. This reduces the possibility of transients occurring at the output(s) due to changes made at the select inputs, even when the '151A outputs are enabled (i.e., strobe low).

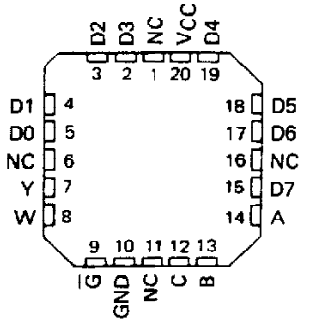
SN54150 . . . J OR W PACKAGE
SN74150 . . . N PACKAGE
(TOP VIEW)



SN54151A, SN54LS151, SN54S151 . . . J OR W PACKAGE
SN74151A . . . N PACKAGE
SN74LS151, SN74S151 . . . D OR N PACKAGE
(TOP VIEW)



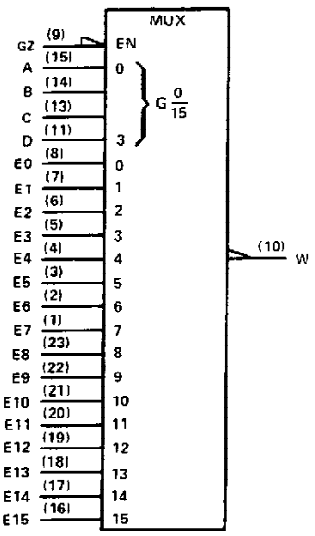
SN54LS151, SN54S151 . . . FK PACKAGE
(TOP VIEW)



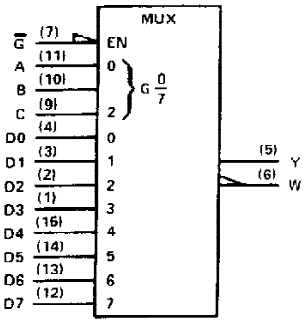
NC - No internal connection

logic symbols†

'150



'151A, 'LS151, 'S151



†These symbols are in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication 617-12.
Pin numbers shown are D, J, N, and W packages.

'150

FUNCTION TABLE

INPUTS					OUTPUT
SELECT				STROBE	
D	C	B	A	$\overline{G}$	W
X	X	X	X	H	H
L	L	L	L	L	$\overline{E0}$
L	L	L	H	L	$\overline{E1}$
L	L	H	L	L	$\overline{E2}$
L	L	H	H	L	$\overline{E3}$
L	H	L	L	L	$\overline{E4}$
L	H	L	H	L	$\overline{E5}$
L	H	H	L	L	$\overline{E6}$
L	H	H	H	L	$\overline{E7}$
H	L	L	L	L	$\overline{E8}$
H	L	L	H	L	$\overline{E9}$
H	L	H	L	L	$\overline{E10}$
H	L	H	H	L	$\overline{E11}$
H	H	L	L	L	$\overline{E12}$
H	H	L	H	L	$\overline{E13}$
H	H	H	L	L	$\overline{E14}$
H	H	H	H	L	$\overline{E15}$

'151A, 'LS151, 'S151

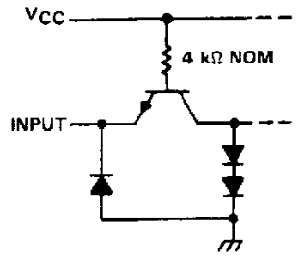
FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE		
C	B	A	$\overline{G}$	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

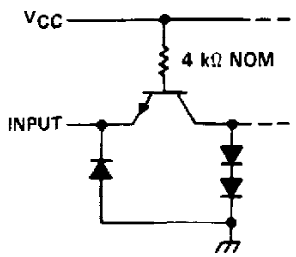
H = high level, L = low level, X = irrelevant
$\overline{E0}$, $\overline{E1}$ . . . $\overline{E15}$ = the complement of the level of the respective E input
D0, D1 . . . D7 = the level of the D respective input

schematics of inputs and outputs

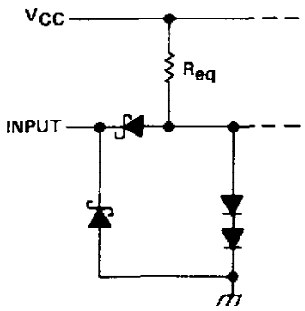
EQUIVALENT OF EACH INPUT OF '150



EQUIVALENT OF EACH INPUT OF '151A

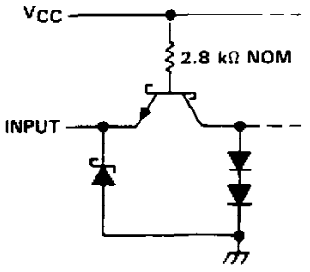


EQUIVALENT OF EACH INPUT OF 'LS151

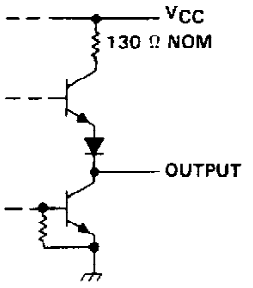


Data select and strobe R_{eq} = 20 kΩ NOM
Data inputs R_{eq} = 17 kΩ NOM

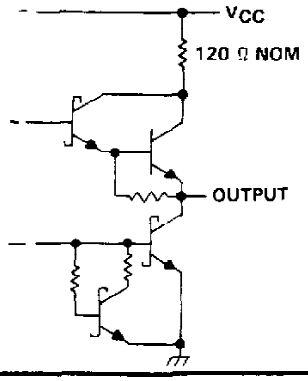
EQUIVALENT OF EACH INPUT OF 'S151



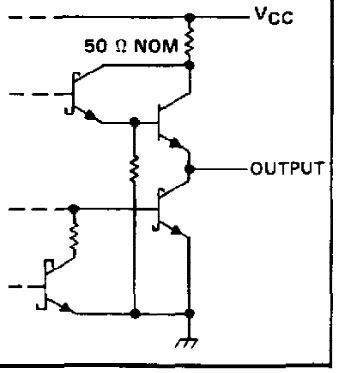
TYPICAL OF ALL OUTPUTS OF '150, '151A



TYPICAL OF ALL OUTPUTS OF 'LS151



TYPICAL OF ALL OUTPUTS OF 'S151



recommended operating conditions

	SN54'			SN74'			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−800			−800	μA
Low-level output current, I_{OL}			16			16	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		'150			'151A			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.8			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −8 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −800 μA		2.4	3.4		2.4	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 16 mA			0.2	0.4		0.2	0.4	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 5.5 V				1			1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.4 V				40			40	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4 V				−1.6			−1.6	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX	SN54'	−20		−55	−20		−55	mA
			SN74'	−18		−55	−18		−55	
I_{CC}	Supply current	V_{CC} = MAX, See Note 3			40	68		29	48	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡ All typical values at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output of the '151A should be shorted at a time.
NOTE 3: I_{CC} is measured with the strobe and data select inputs at 4.5 V, all other inputs and outputs open.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'150			'151A			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	A, B, or C (4 levels)	Y	C_L = 15 pF, R_L = 400 Ω, See Note 4					25	38	ns
t_{PHL}								25	38	
t_{PLH}	A, B, C, or D (3 levels)	W			23	35		17	26	ns
t_{PHL}					22	33		19	30	
t_{PLH}	Strobe $\overline{G}$	Y						21	33	ns
t_{PHL}								22	33	
t_{PLH}	Strobe $\overline{G}$	W			15.5	24		14	21	ns
t_{PHL}					21	30		15	23	
t_{PLH}	D0 thru D7	Y						13	20	ns
t_{PHL}								18	27	
t_{PLH}	E0 thru E15, or D0 thru D7	W			8.5	14		8	14	ns
t_{PHL}					13	20		8	14	

¶ t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
NOTE 4: Load circuits and voltage waveforms are shown in Section 1.

recommended operating conditions

	SN54LS151			SN74LS151			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−400			−400	μA
Low-level output current, I_{OL}			4			8	mA
Operating free-air temperature, T_A	−55		125	0		70	C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		SN54LS151			SN74LS151			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.7			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL}max, I_{OH} = −400 μA		2.5	3.4		2.7	3.4		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL}max	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
			I_{OL} = 8 mA					0.35	0.5	
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				20			20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−20		−100	−20		−100	mA
I_{CC}	Supply current	V_{CC} = MAX, Outputs open, All inputs at 4.5 V			6.0	10		6.0	10	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time and duration of short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A 25°C

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	A, B, or C (4 levels)	Y	C_L = 15 pF, R_L = 2 kΩ, See Note 4		27	43	ns
t_{PHL}					18	30	
t_{PLH}	A, B, or C (3 levels)	W			14	23	ns
t_{PHL}					20	32	
t_{PLH}	Strobe $\overline{G}$	Y			26	42	ns
t_{PHL}					20	32	
t_{PLH}	Strobe $\overline{G}$	W			15	24	ns
t_{PHL}					18	30	
t_{PLH}	Any D	Y			20	32	ns
t_{PHL}					16	26	
t_{PLH}	Any D	W			13	21	ns
t_{PHL}					12	20	

¶ t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
NOTE 4: Load circuits and voltage waveforms are shown in Section 1.

recommended operating conditions

	SN54S151			SN74S151			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}	4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}			−1			−1	mA
Low-level output current, I_{OL}			20			20	mA
Operating free-air temperature, T_A	−55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		MIN	TYP‡	MAX	UNIT
V_{IH}	High-level input voltage			2			V
V_{IL}	Low-level input voltage					0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.2	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OH} = −1 mA	SN54S151	2.5	3.4		V
			SN74S151	2.7	3.4		
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = 0.8 V, I_{OL} = 20 mA				0.5	V
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 5.5 V				1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				50	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.5 V				−2	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−40		−100	mA
I_{CC}	Supply current	V_{CC} = MAX, All inputs at 4.5 V, All outputs open			45	70	mA

†For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions for the applicable device type.
‡All typical values are at V_{CC} = 5 V, T_A = 25°C.
§Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A 25°C

PARAMETER¶	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	A, B, or C (4 levels)	Y	C_L = 15 pF, R_L = 280 kΩ, See Note 4		12	18	ns
t_{PHL}					12	18	
t_{PLH}	A, B, or C (3 levels)	W			10	15	ns
t_{PHL}					9	13.5	
t_{PLH}	Any D	Y			8	12	ns
t_{PHL}					8	12	
t_{PLH}	Any D	W			4.5	7	ns
t_{PHL}					4.5	7	
t_{PLH}	Strobe $\overline{G}$	Y			11	16.5	ns
t_{PHL}					12	18	
t_{PLH}	Strobe $\overline{G}$	W			9	13	ns
t_{PHL}					8.5	12	

¶t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
NOTE 4: Load circuits and voltage waveforms are shown in Section 1.

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead finish/ Ball material (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
76010012A	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	76010012A SNJ54LS 151FK	Samples
7601001EA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	7601001EA SNJ54LS151J	Samples
7601001EA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	7601001EA SNJ54LS151J	Samples
JM38510/07901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 07901BEA	Samples
JM38510/07901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 07901BFA	Samples
JM38510/30901B2A	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901B2A	Samples
JM38510/30901B2A	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901B2A	Samples
JM38510/30901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BEA	Samples
JM38510/30901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BEA	Samples
JM38510/30901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BFA	Samples
JM38510/30901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BFA	Samples
M38510/07901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 07901BEA	Samples
M38510/07901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 07901BFA	Samples
M38510/30901B2A	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901B2A	Samples
M38510/30901B2A	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901B2A	Samples
M38510/30901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BEA	Samples

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead finish/ Ball material (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
M38510/30901BEA	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BEA	Samples
M38510/30901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BFA	Samples
M38510/30901BFA	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	JM38510/ 30901BFA	Samples
SN54LS151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	SN54LS151J	Samples
SN54LS151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	SN54LS151J	Samples
SN54S151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	SN54S151J	Samples
SN74LS151DR	ACTIVE	SOIC	D	16	2500	RoHS & Green	NIPDAU	Level-1-260C-UNLIM	0 to 70	LS151	Samples
SN74LS151DR	ACTIVE	SOIC	D	16	2500	RoHS & Green	NIPDAU	Level-1-260C-UNLIM	0 to 70	LS151	Samples
SN74LS151N	ACTIVE	PDIP	N	16	25	RoHS & Green	NIPDAU	N / A for Pkg Type	0 to 70	SN74LS151N	Samples
SN74LS151N	ACTIVE	PDIP	N	16	25	RoHS & Green	NIPDAU	N / A for Pkg Type	0 to 70	SN74LS151N	Samples
SN74LS151NE4	ACTIVE	PDIP	N	16	25	RoHS & Green	NIPDAU	N / A for Pkg Type	0 to 70	SN74LS151N	Samples
SN74LS151NE4	ACTIVE	PDIP	N	16	25	RoHS & Green	NIPDAU	N / A for Pkg Type	0 to 70	SN74LS151N	Samples
SN74LS151NSR	ACTIVE	SO	NS	16	2000	RoHS & Green	NIPDAU	Level-1-260C-UNLIM	0 to 70	74LS151	Samples
SN74LS151NSR	ACTIVE	SO	NS	16	2000	RoHS & Green	NIPDAU	Level-1-260C-UNLIM	0 to 70	74LS151	Samples
SNJ54LS151FK	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	76010012A SNJ54LS 151FK	Samples
SNJ54LS151FK	ACTIVE	LCCC	FK	20	55	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	76010012A SNJ54LS 151FK	Samples
SNJ54LS151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	7601001EA SNJ54LS151J	Samples
SNJ54LS151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	7601001EA SNJ54LS151J	Samples

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead finish/ Ball material (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
SNJ54S151J	ACTIVE	CDIP	J	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	SNJ54S151J	Samples
SNJ54S151W	ACTIVE	CFP	W	16	25	Non-RoHS & Green	SNPB	N / A for Pkg Type	-55 to 125	SNJ54S151W	Samples

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) **RoHS:** TI defines "RoHS" to mean semiconductor products that are compliant with the current EU RoHS requirements for all 10 RoHS substances, including the requirement that RoHS substance do not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, "RoHS" products are suitable for use in specified lead-free processes. TI may reference these types of products as "Pb-Free".
RoHS Exempt: TI defines "RoHS Exempt" to mean products that contain lead but are compliant with EU RoHS pursuant to a specific EU RoHS exemption.
Green: TI defines "Green" to mean the content of Chlorine (Cl) and Bromine (Br) based flame retardants meet JS709B low halogen requirements of <=1000ppm threshold. Antimony trioxide based flame retardants must also meet the <=1000ppm threshold requirement.

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead finish/Ball material - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead finish/Ball material values may wrap to two lines if the finish value exceeds the maximum column width.

OTHER QUALIFIED VERSIONS OF SN54LS151, SN74LS151 :

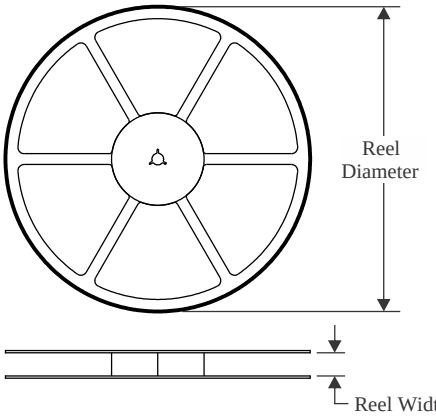
- Catalog : SN74LS151
- Military : SN54LS151

NOTE: Qualified Version Definitions:

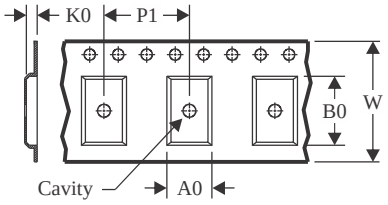
- Catalog - TI's standard catalog product
- Military - QML certified for Military and Defense Applications

TAPE AND REEL INFORMATION

REEL DIMENSIONS

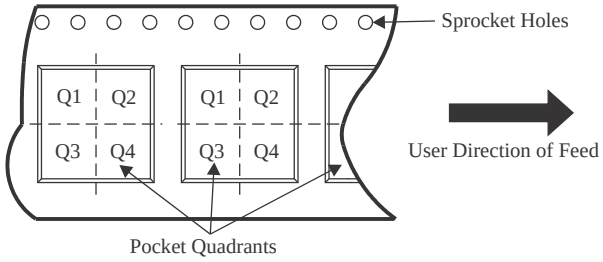


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

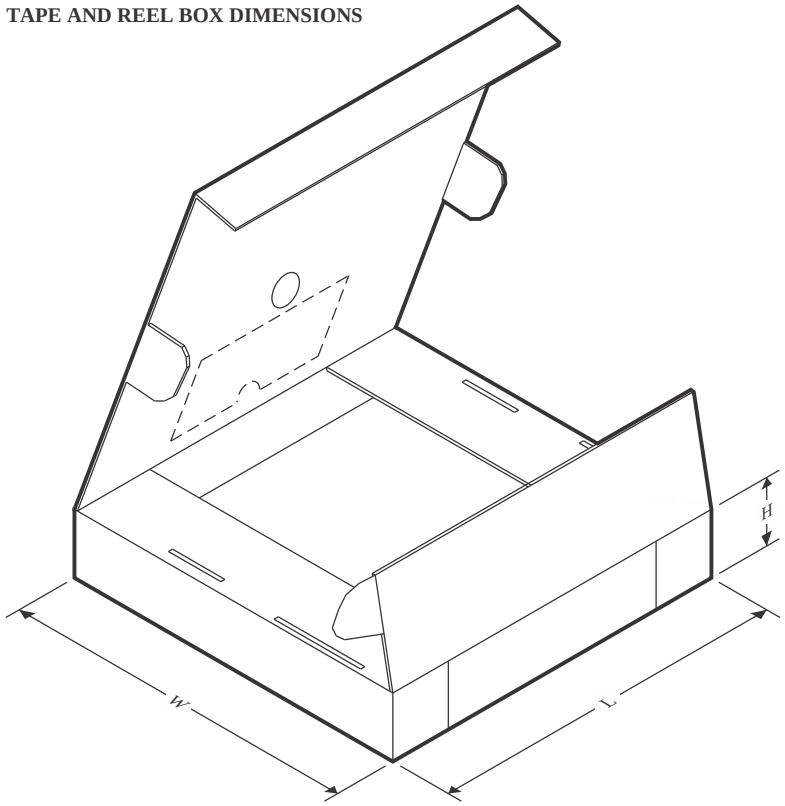
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
SN74LS151DR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
SN74LS151NSR	SO	NS	16	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1

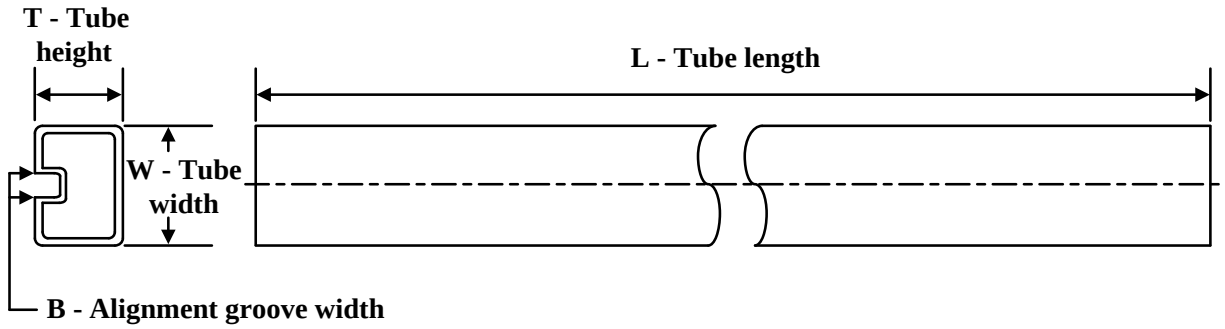
TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

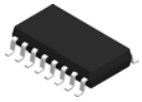
Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
SN74LS151DR	SOIC	D	16	2500	340.5	336.1	32.0
SN74LS151NSR	SO	NS	16	2000	356.0	356.0	35.0

TUBE



*All dimensions are nominal

Device	Package Name	Package Type	Pins	SPQ	L (mm)	W (mm)	T (µm)	B (mm)
76010012A	FK	LCCC	20	55	506.98	12.06	2030	NA
JM38510/07901BFA	W	CFP	16	25	506.98	26.16	6220	NA
JM38510/30901B2A	FK	LCCC	20	55	506.98	12.06	2030	NA
JM38510/30901BFA	W	CFP	16	25	506.98	26.16	6220	NA
M38510/07901BFA	W	CFP	16	25	506.98	26.16	6220	NA
M38510/30901B2A	FK	LCCC	20	55	506.98	12.06	2030	NA
M38510/30901BFA	W	CFP	16	25	506.98	26.16	6220	NA
SN74LS151N	N	PDIP	16	25	506	13.97	11230	4.32
SN74LS151N	N	PDIP	16	25	506	13.97	11230	4.32
SN74LS151NE4	N	PDIP	16	25	506	13.97	11230	4.32
SN74LS151NE4	N	PDIP	16	25	506	13.97	11230	4.32
SNJ54LS151FK	FK	LCCC	20	55	506.98	12.06	2030	NA

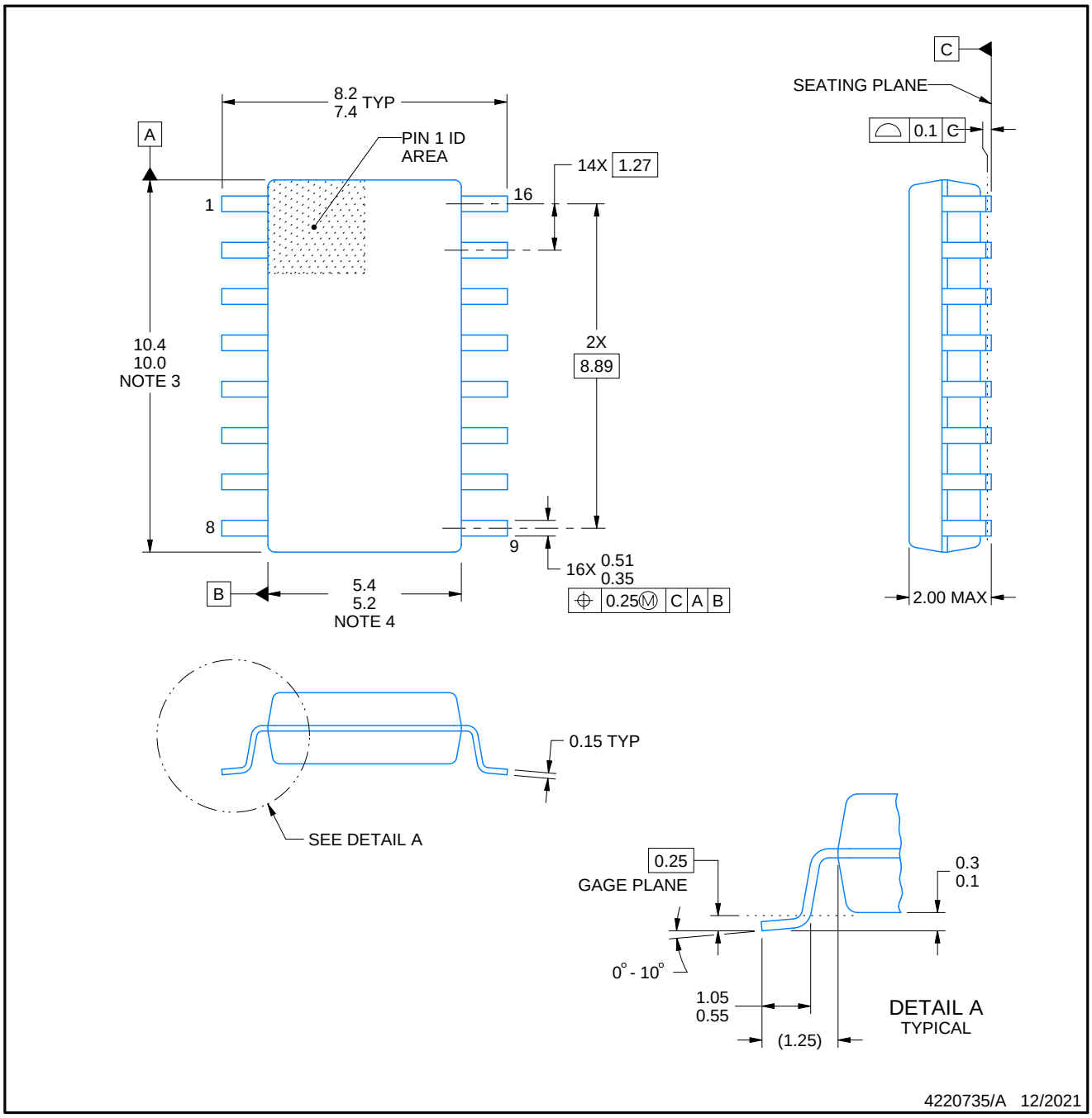


PACKAGE OUTLINE

NS0016A

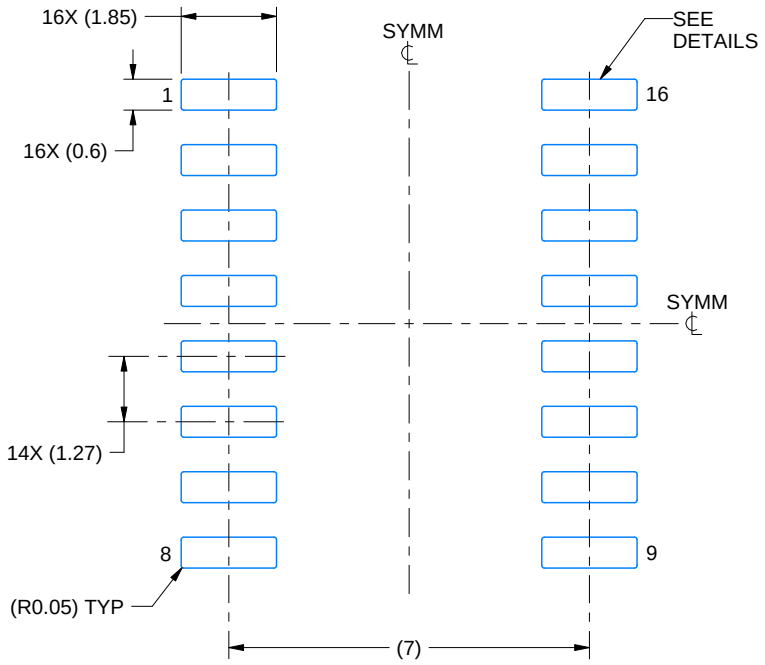
SOP - 2.00 mm max height

SOP

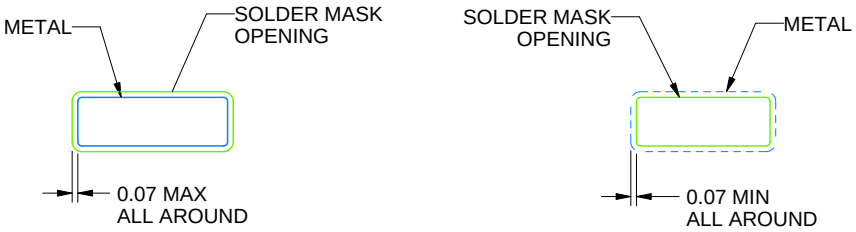


NOTES:

1. All linear dimensions are in millimeters. Dimensions in parenthesis are for reference only. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. This dimension does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.15 mm, per side.
4. This dimension does not include interlead flash. Interlead flash shall not exceed 0.25 mm, per side.



LAND PATTERN EXAMPLE
SCALE:7X



NON SOLDER MASK
DEFINED

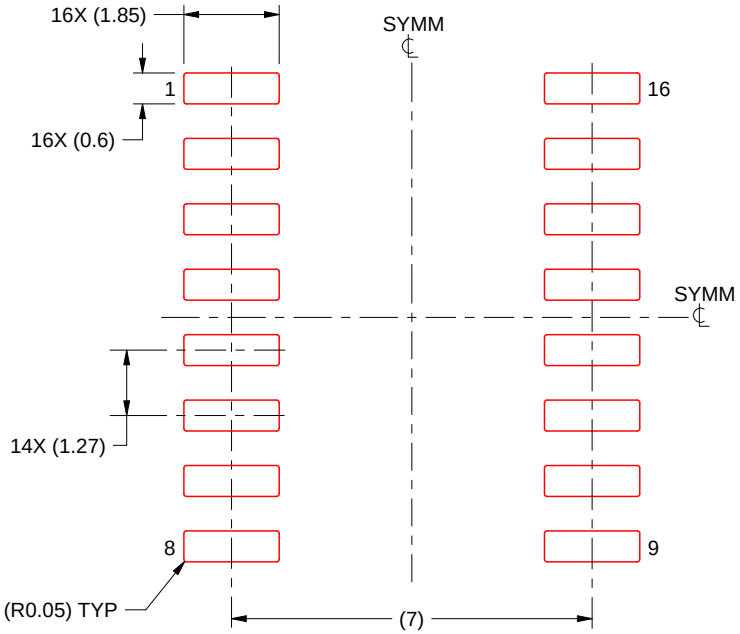
SOLDER MASK
DEFINED

SOLDER MASK DETAILS

4220735/A 12/2021

NOTES: (continued)

5. Publication IPC-7351 may have alternate designs.
6. Solder mask tolerances between and around signal pads can vary based on board fabrication site.



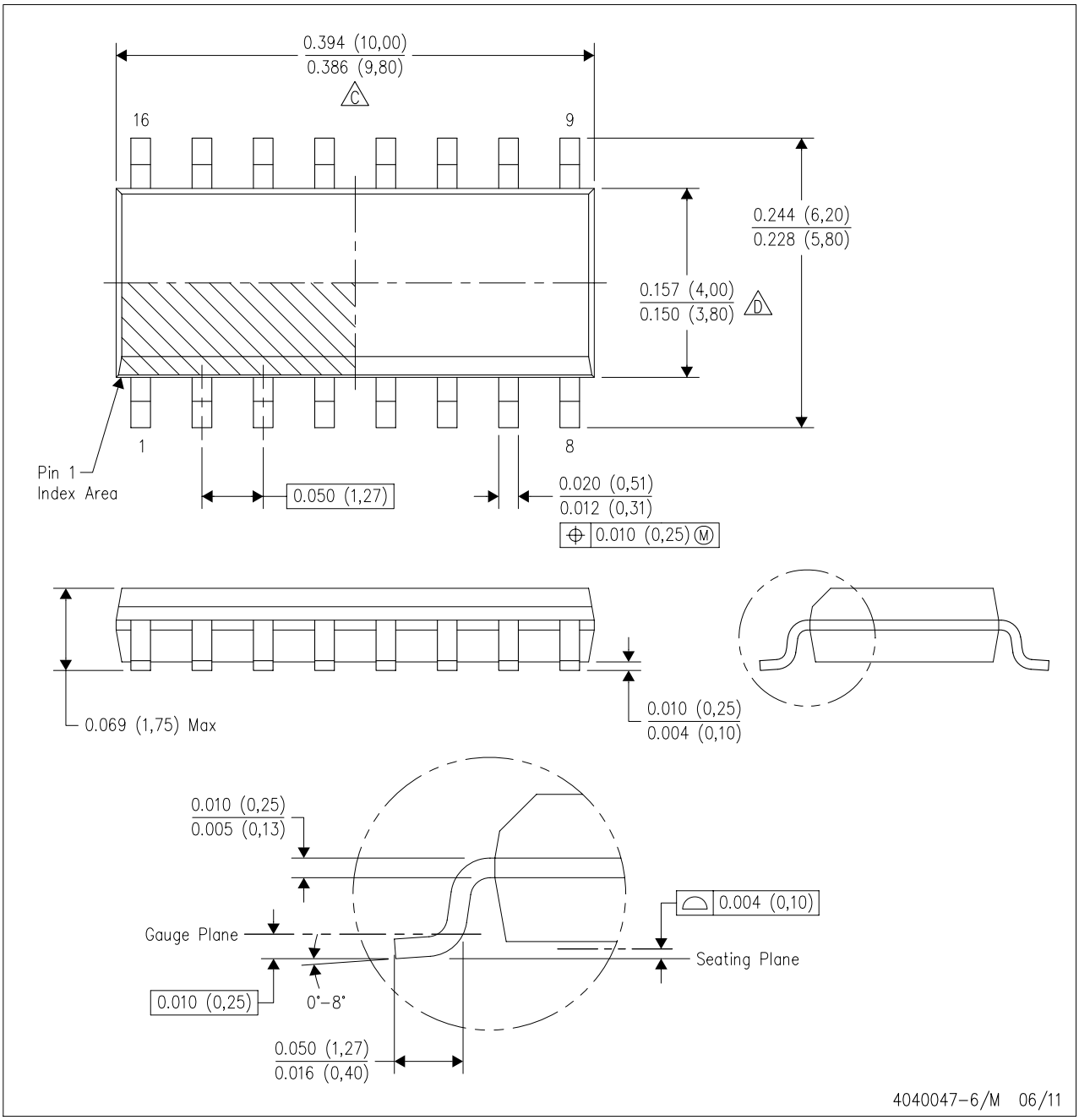
SOLDER PASTE EXAMPLE
BASED ON 0.125 mm THICK STENCIL
SCALE:7X

4220735/A 12/2021

NOTES: (continued)

7. Laser cutting apertures with trapezoidal walls and rounded corners may offer better paste release. IPC-7525 may have alternate design recommendations.
8. Board assembly site may have different recommendations for stencil design.

D (R–PDSO–G16) PLASTIC SMALL OUTLINE

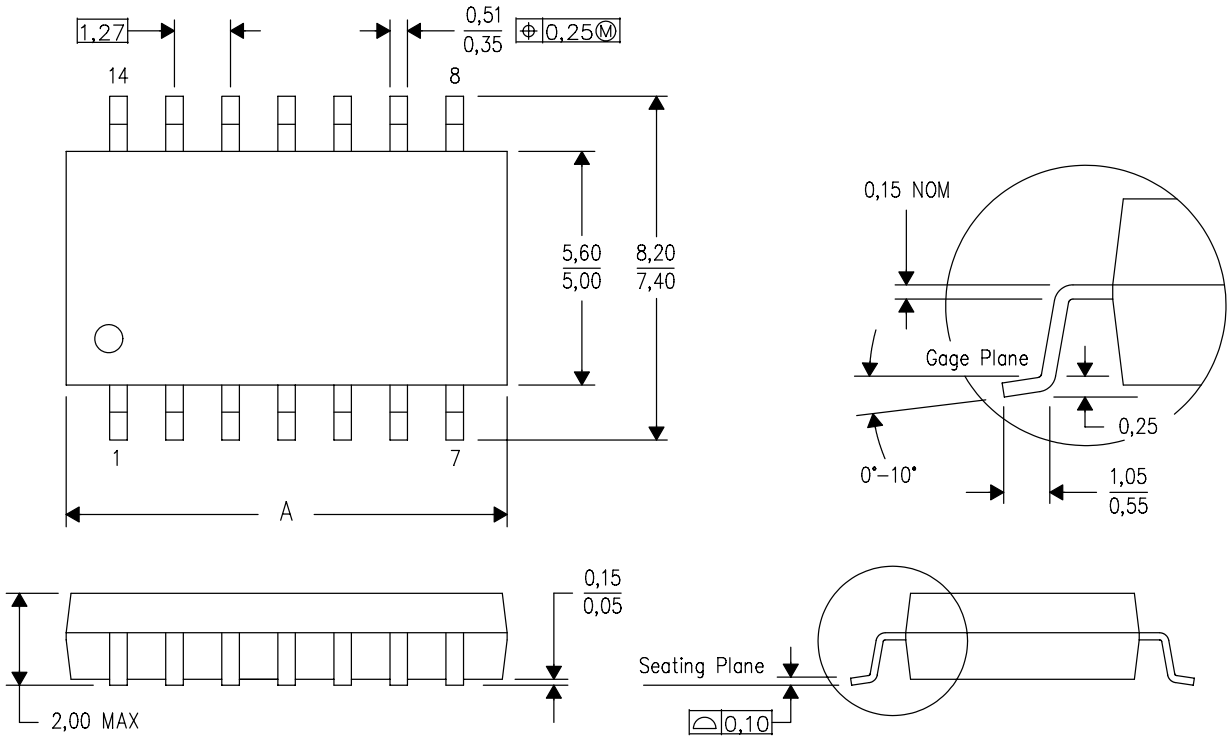


NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.006 (0,15) each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0.017 (0,43) each side.
E. Reference JEDEC MS–012 variation AC.

MECHANICAL DATA

NS (R-PDSO-G)** **PLASTIC SMALL-OUTLINE PACKAGE**

14-PINS SHOWN

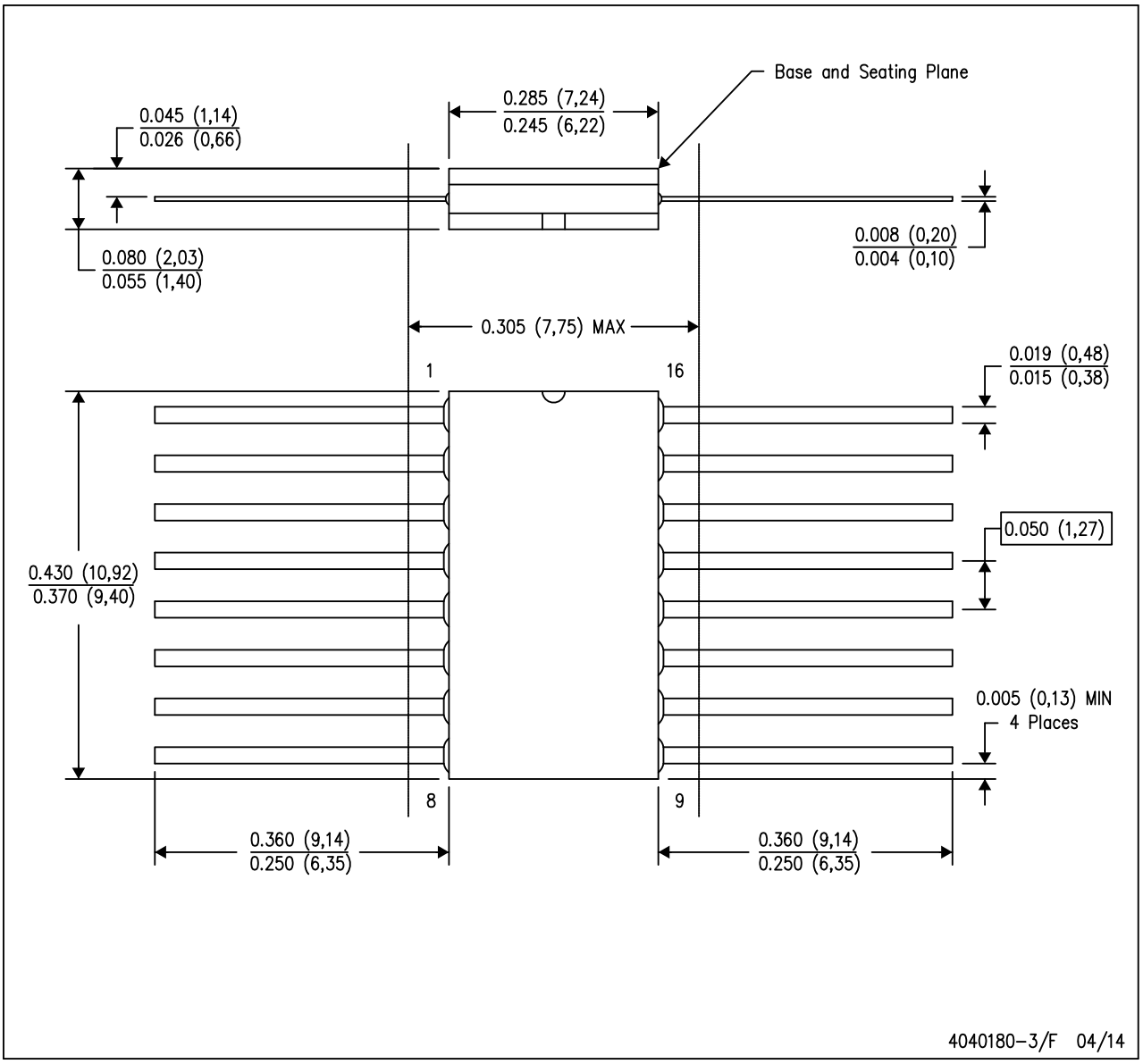


DIM \ PINS **	14	16	20	24
A MAX	10,50	10,50	12,90	15,30
A MIN	9,90	9,90	12,30	14,70

4040062/C 03/03

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.

W (R–GDFP–F16) CERAMIC DUAL FLATPACK



NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package can be hermetically sealed with a ceramic lid using glass frit.
D. Index point is provided on cap for terminal identification only.
E. Falls within MIL STD 1835 GDFP2–F16

GENERIC PACKAGE VIEW

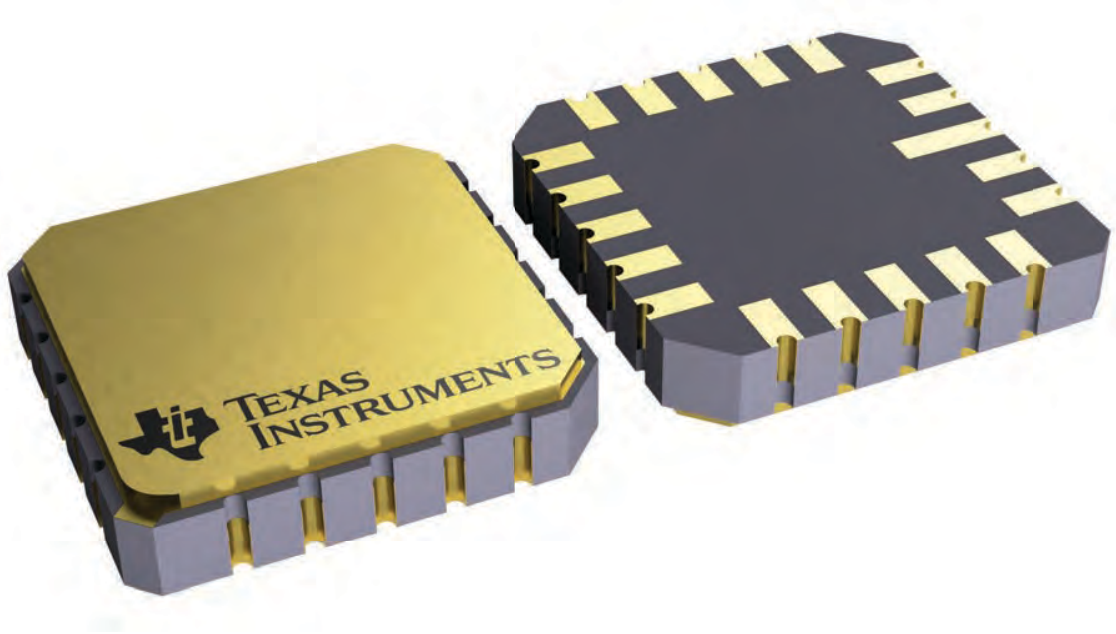
FK 20

LCCC - 2.03 mm max height

8.89 x 8.89, 1.27 mm pitch

LEADLESS CERAMIC CHIP CARRIER

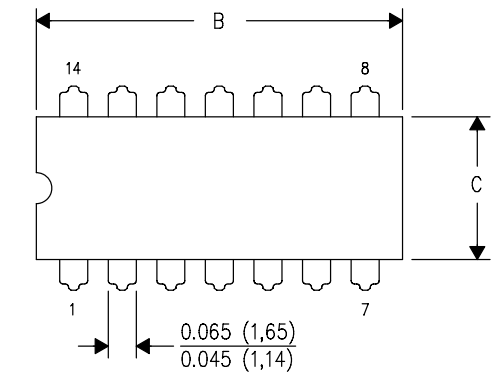
This image is a representation of the package family, actual package may vary.
Refer to the product data sheet for package details.



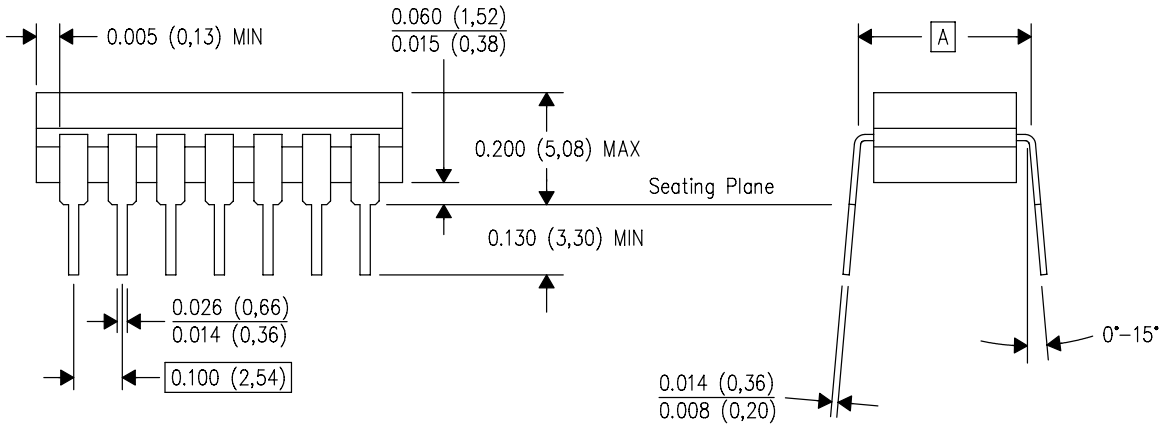
J (R–GDIP–T**)

CERAMIC DUAL IN–LINE PACKAGE

14 LEADS SHOWN



DIM \ PINS **	14	16	18	20
A	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC
B MAX	0.785 (19,94)	.840 (21,34)	0.960 (24,38)	1.060 (26,92)
B MIN	—	—	—	—
C MAX	0.300 (7,62)	0.300 (7,62)	0.310 (7,87)	0.300 (7,62)
C MIN	0.245 (6,22)	0.245 (6,22)	0.220 (5,59)	0.245 (6,22)



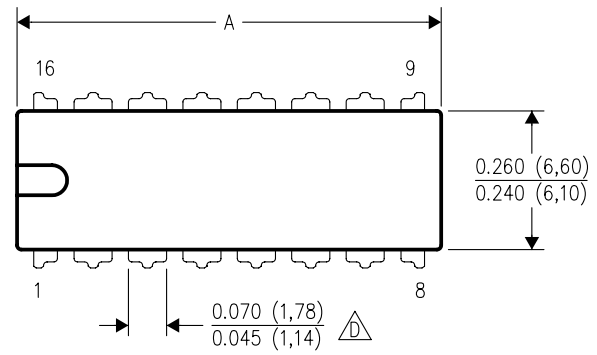
4040083/F 03/03

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package is hermetically sealed with a ceramic lid using glass frit.
D. Index point is provided on cap for terminal identification only on press ceramic glass frit seal only.
E. Falls within MIL STD 1835 GDIP1–T14, GDIP1–T16, GDIP1–T18 and GDIP1–T20.

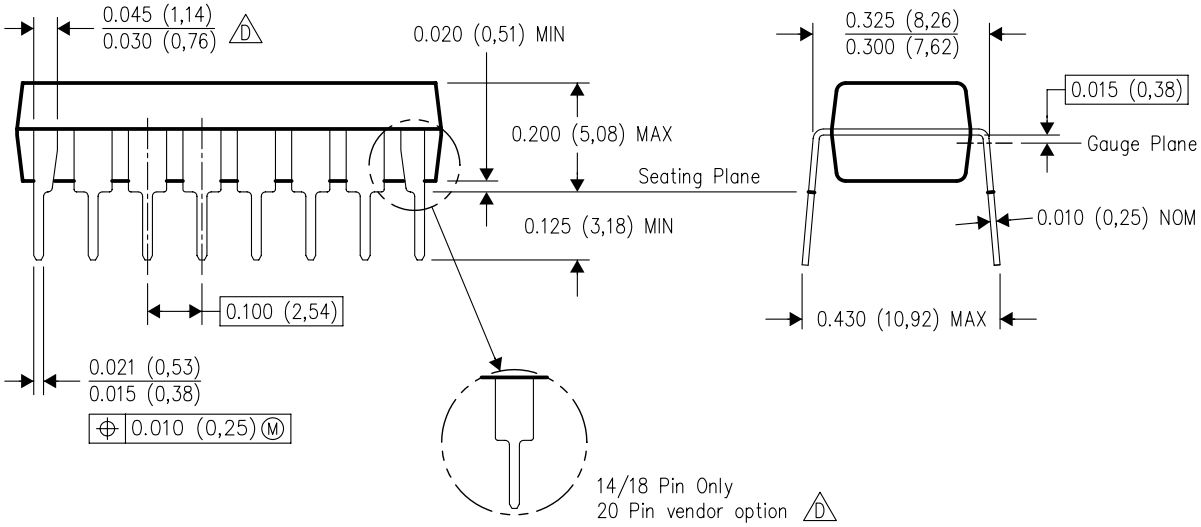
N (R–PDIP–T**)

PLASTIC DUAL–IN–LINE PACKAGE

16 PINS SHOWN



DIM \ PINS **	14	16	18	20
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS–001 VARIATION (C)	AA	BB	AC	AD



4040049/E 12/2002

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS–001, except 18 and 20 pin minimum body length (Dim A).
D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

B.7 Datasheet: LTR-390UV-01

LITEON®
OPTOELECTRONICS

Optical Sensor

Product Data Sheet

LTR-390UV-01

Spec No.: DS86-2015-0004

Effective Date: 02/03/2016

Revision: -

LITE-ON DCC

RELEASE

BNS-OD-FC001/A4

LITE-ON Technology Corp. / Optoelectronics
No.90,Chien 1 Road, Chung Ho, New Taipei City 23585, Taiwan, R.O.C.
Tel: 886-2-2222-6181 Fax: 886-2-2221-1948 / 886-2-2221-0660
http://www.liteon.com/opto

OPTICAL SENSOR
LTR-390UV-01

Description

The LTR-390UV-01 is an integrated low voltage I2C ambient light sensor (ALS) and ultraviolet light sensor (UVS) in a single miniature 2x2mm chipled lead-free surface mount package.

This sensor converts light intensity to a digital output signal capable of direct I2C interface. It provides a linear ALS response over a wide dynamic range, and is well suited to applications under high ambient brightness.

The sensor has a programmable interrupt with hysteresis to response to events and that removes the need to poll the sensor for a reading which improves system efficiency. This CMOS design and factory-set one time trimming capability ensure minimal sensor-to-sensor variations for ease of manufacturability to the end customers.

Application

- Identifying the UV index in ambient light helps people to effectively protect themselves from sunburns, cancer or eye damage.
- To control brightness and color of the display panel in mobile, computing, and consumer devices.

Features

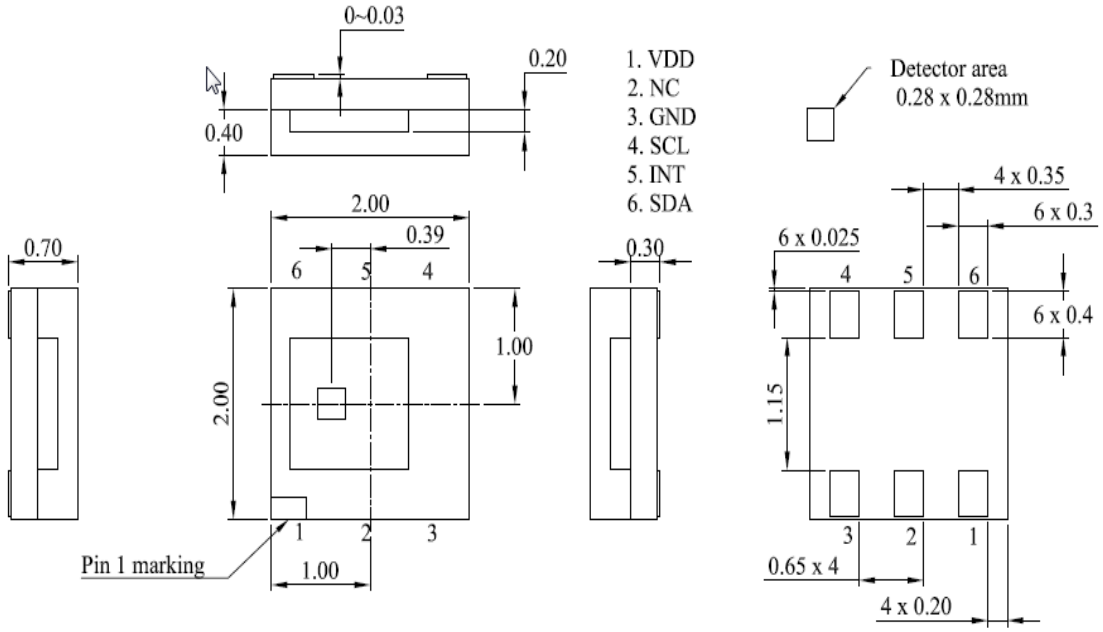
- I^2C interface capable of Standard mode @100kHz or Fast mode @400kHz communication; 1.8V logic compatible
- Ambient Light / Ultraviolet light (UVS) Technology in one ultra-small 2x2mm ChipLED package
- Very low power consumption with sleep mode capability
- Operating voltage ranges: 1.7V to 3.6V
- Operating temperature ranges: -40 to +85 ºC
- Built-in temperature compensation circuit
- Programmable interrupt function for ALS , UVS with upper and lower thresholds
- RoHS and Halogen free compliant
- **UVS/ALS Features**
  - 13 to 20 bits effective resolution
  - Wide dynamic range of 1:18,000,000 with linear response
  - Close to human eye spectral response
  - Automatic rejection for 50Hz/60Hz lighting flicker

Ordering Information

Part Number	Packaging Type	Package	Quantity
LTR-390UV-01	Tape and Reel	6-pin chipled package	2500

OPTICAL SENSOR
LTR-390UV-01

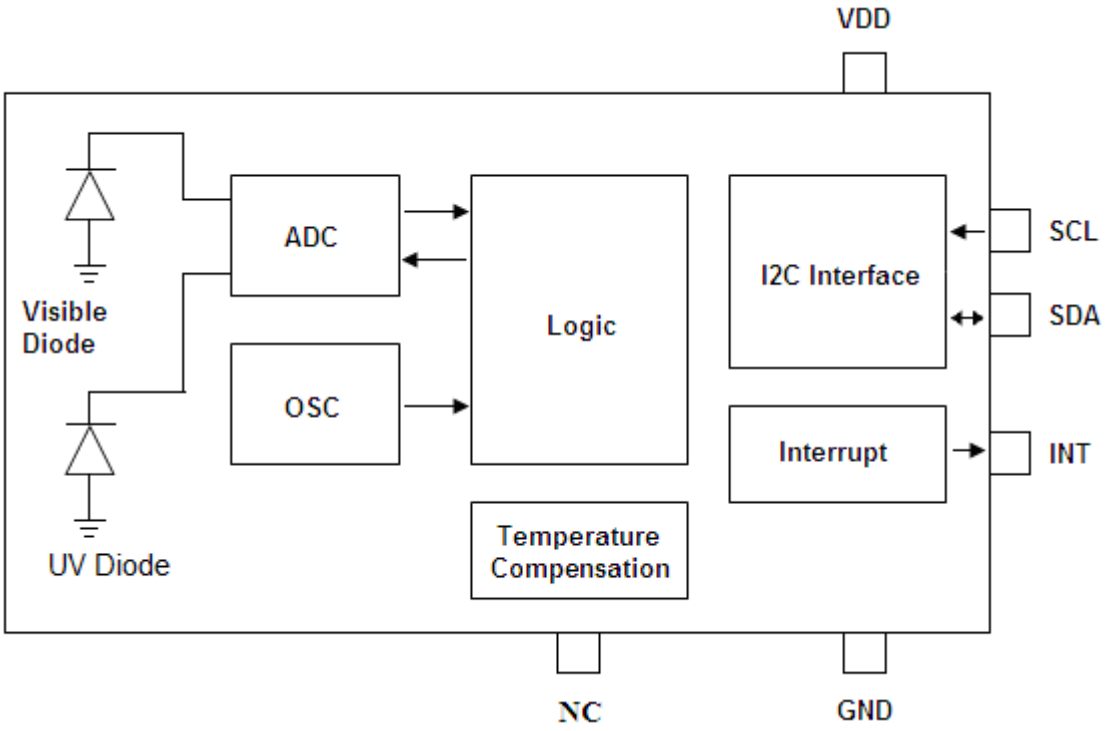
1. Outline Dimensions



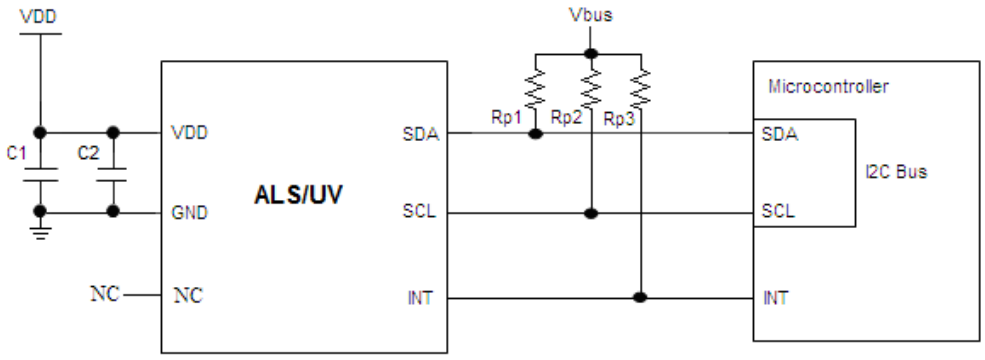
1. All dimensions in mm
2. Tolerances is +/-0.2
3. LTC reserve the right to to change the drawing till final datasheet release

2. Functional Block Diagram

LTR-390UV-01 contains 2 integrated photodiodes (ALS/UVS) for respective photocurrent measurements. The photodiode currents are converted to digital values by ADCs. The sensor also includes some peripheral circuits such as an internal oscillator and voltage reference.



3. Application Circuit



I/O Pins Configuration Table

Pin	I/O Type	Symbol	Description
1		VDD	Power Supply Voltage
2		NC	No connection to this pin
3		GND	Ground
4	I	SCL*	I^2C serial clock. This pin is an open drain input.
5	O	INT*	Level Interrupt Pin. This pin is an open drain output.
6	I/O	SDA*	I^2C serial data. This pin is an open drain input / output.

** Note: For noisy environment, add 10pF capacitor from signal to GND for additional noise filtering.*

Recommended Application Circuit Components

Component	Recommended Value
Rp1, Rp2, Rp3 [1]	1 kΩ to 10 kΩ
C1	0.1uF
C2	4.7uF

Notes:

[1] Selection of pull-up resistors value is dependent on bus capacitance values. For more details, please refer to I2C Specifications: http://www.nxp.com/documents/user_manual/UM10204.pdf

4. Rating and Specification

4.1. Absolute Maximum Rating at Ta=25°C

Parameter	Symbol	Min.	Max.	Unit
Supply Voltage	VDD		4.0	V
Digital Voltage Range	SCL, SDA, INT	-0.5	4.0	V
Storage Temperature	T_{stg}		-45 to 95	°C
Max. Input Current	SCL,SDA,INT	-100	100	mA
Electrostatic Discharge Protection (Human Body Model JESD22-A114)	V_{HBM}		2000	V

Note: Exceeding these ratings could cause damage to the sensor. All voltages are with respect to ground. Currents are positive into, negative out of the specified terminal.

4.2. Recommended Operating Conditions

Description	Symbol	Min.	Typ.	Max.	Unit
Supply Voltage	VDD	1.7		3.6	V
Interface signal input high	$V_{I2Chigh}$	1.5		VDD	V
Interface signal input low	V_{I2Clow}	0		0.4	V
Operating Temperature	T_{ope}	-40		85	°C

4.3. Electrical Specifications (VDD = 2.8V, Ta=25°C, unless otherwise noted)

Parameter	Min.	Typ.	Max.	Unit	Condition
ALS Active Mode Current		110		uA	Max. duty cycle, Vdd=2.8V, Gain 3x
UVS Active Mode Current		100		uA	Max. duty cycle, Vdd=2.8V
Standby Current		1		uA	Standby / Sleep Mode
Wakeup Time from Standby		5	10	ms	From Standby to Active mode where measurement can start

4.4. Characteristics Ambient Light

Parameter	Min.	Typ.	Max.	Unit	Condition
ALS Output Resolution	13	18	20	Bit	Programmable for 13,16,17,18,19, 20 bit
Dark Level Count		0	5	count	0 Lux, T_{ope}=25°C, 18-bit range
Calibrated Lux Error In Gain Range 3	-10		10	%	White LED,5000K, T_{ope}=+25°C
ALS Accuracy	-25		25	%	Across different light sources
50/60 Hz flicker noise error	-5		5	%	

4.5. Characteristics UVS Sensor

Parameter	Min.	Typ.	Max.	Unit	Condition
UVS Output Resolution	13	18	20	Bit	Programmable for 13,16,17,18,19,20 bit
UV Count		160		count	UV LED 310nm, T_{ope}=25°C, 18-bit Gain range =18, Irradiance =70uW/cm2
UV Sensitivity		2300		Counts/UVI	Gain range = 18, 20-bit
UVI accuracy	-20		20	%	Gain range = 18, 20-bit for UVI>5
	-1		1	UVI	UVI<5

4.6. Typical Device Parameter

(VDD = 2.8V, Ta=25°C, Default power-up settings, unless otherwise noted)

ALS Spectral Response

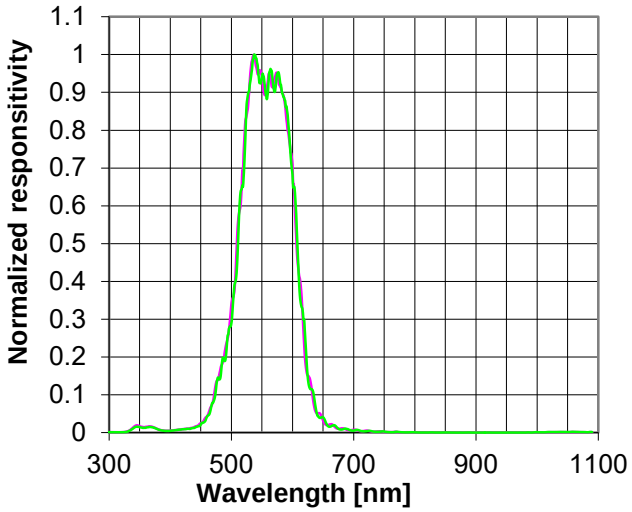


Figure 4.1 Spectral Response of ALS

UV Response

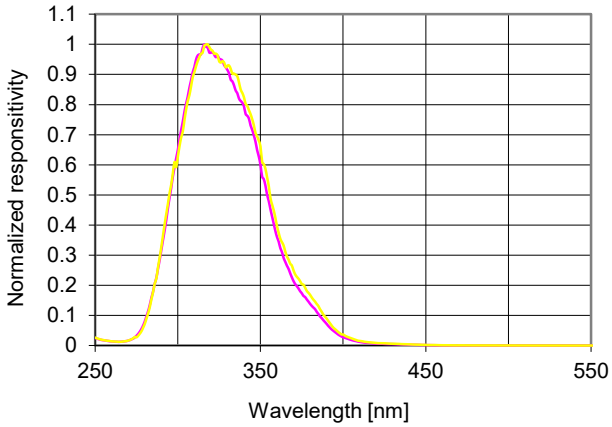


Figure 4.2 UV Spectral Response

Angular of Incidence

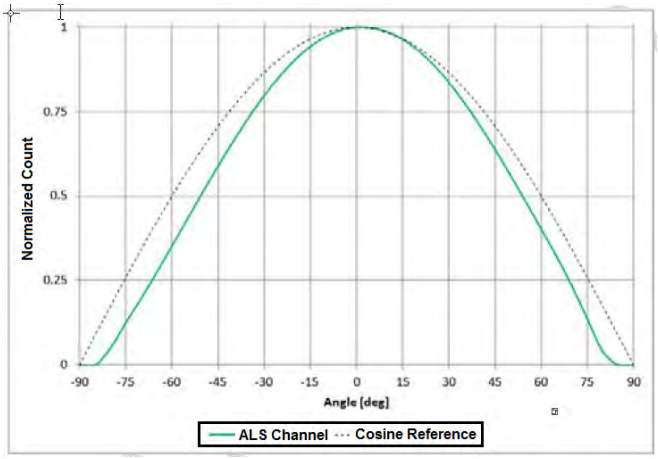


Figure 4.3 ALS Sensitivity vs. Angular of Incidence

UV Count vs Source Irradiance

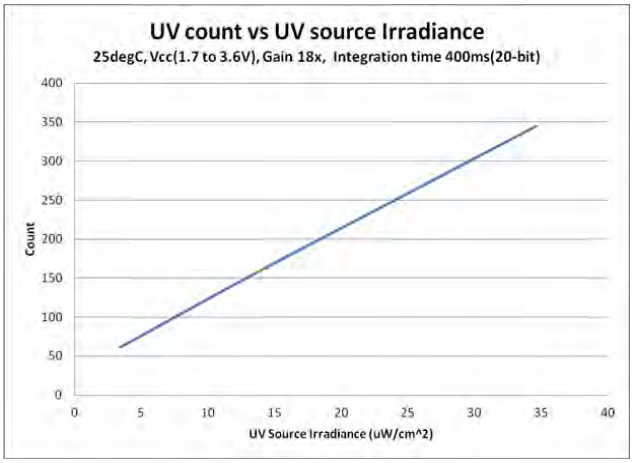


Figure 4.4 UV Count vs Source Irradiance

UV Count vs Gain

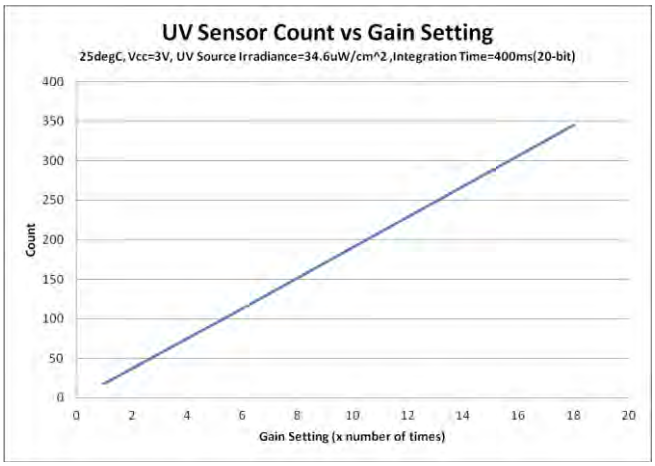


Figure 4.5 UV Count vs Gain

Sensor UVI vs Ref UVI
(Gain 18x, 20-Bit, Sensitivity=2300 counts/UVI)

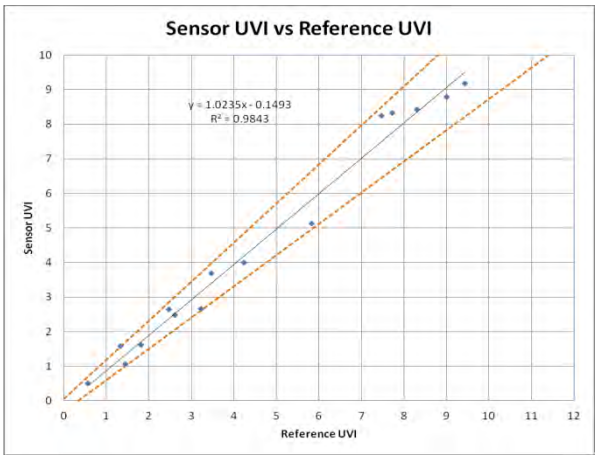
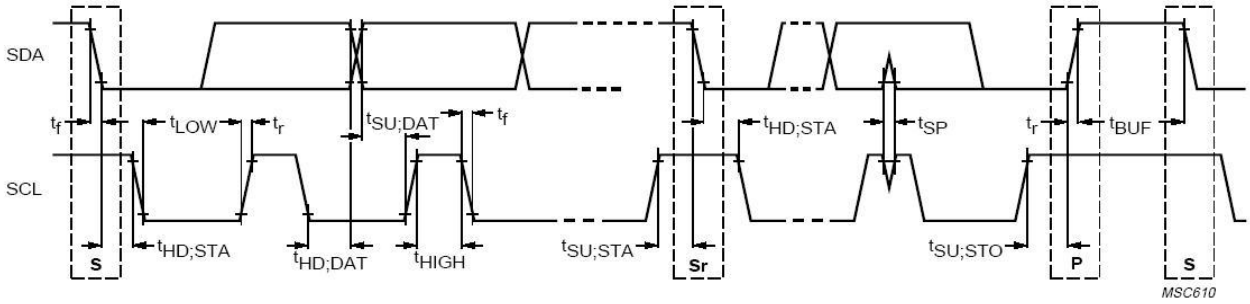


Figure 4.6 Sensor UVI vs Reference UVI
(Reference meter UVMICROLOG by sglux)

4.8. AC Electrical Characteristics

All specifications are at VBus = 1.7V, T_{ope} = 25°C, unless otherwise noted.

Parameter	Symbol	Standard (Min)	Fast (Min)	Unit
SCL clock frequency	f_{SCL}	100	400	KHz
Bus free time between a STOP and START condition	t_{BUF}	4.7		us
Hold time (repeated) START condition. After this period, the first clock pulse is generated	$t_{HD;STA}$	4		us
LOW period of the SCL clock	t_{LOW}	4.7		us
HIGH period of the SCL clock	t_{HIGH}	4		us
Set-up time for a repeated START condition	$t_{SU;STA}$	4.7		us
Set-up time for STOP condition	$t_{SU;STO}$	4		us
Rise time of both SDA and SCL signals	t_r	30	300	ns
Fall time of both SDA and SCL signals	t_f	30	300	ns
Data hold time	$t_{HD;DAT}$	0		us
Data setup time	$t_{SU;DAT}$	100	100	ns
Pulse width of spikes which must be suppressed by the input filter	t_{SP}	0	50	ns

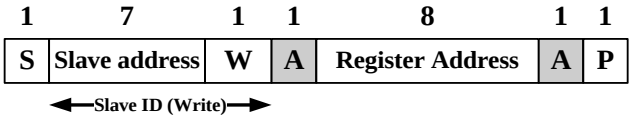


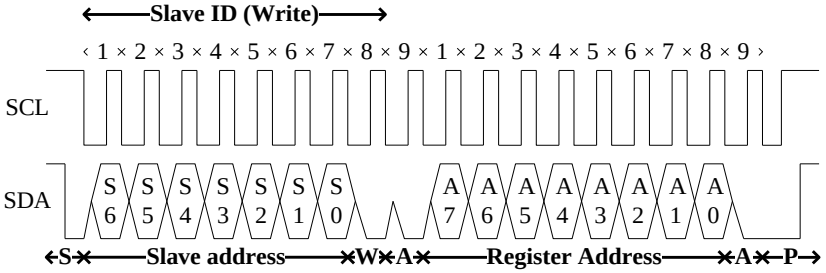
Definition of timing for I²C bus

5. Principle of Operation

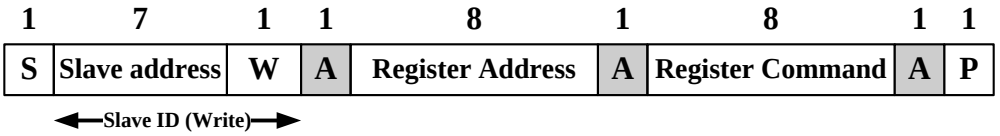
5.1 I2C Protocol

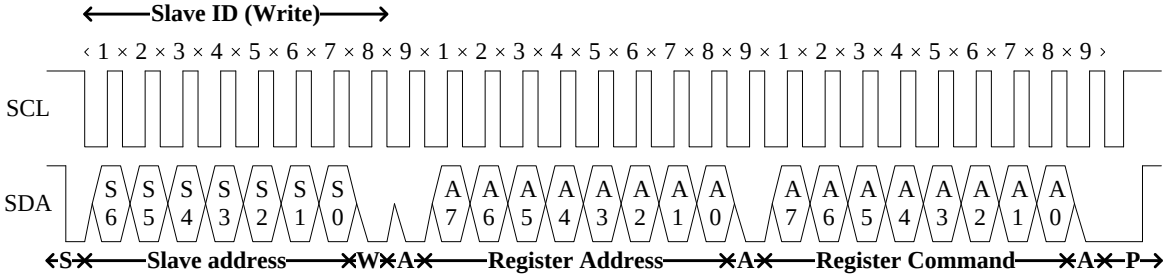
I. I2C Write Protocol



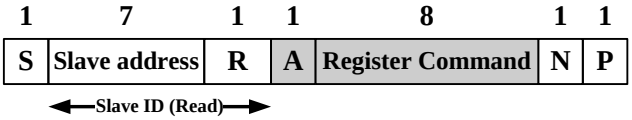


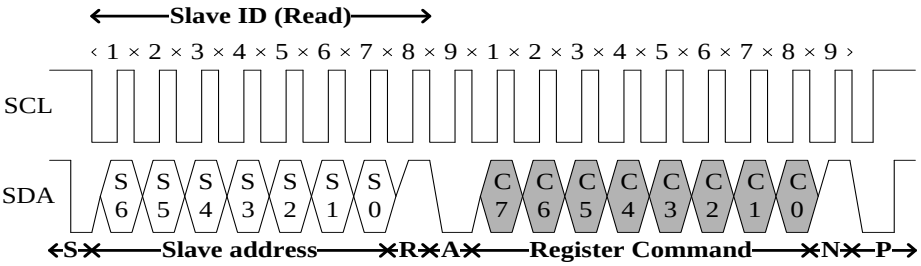
II. I2C Write (Block Write) Protocol



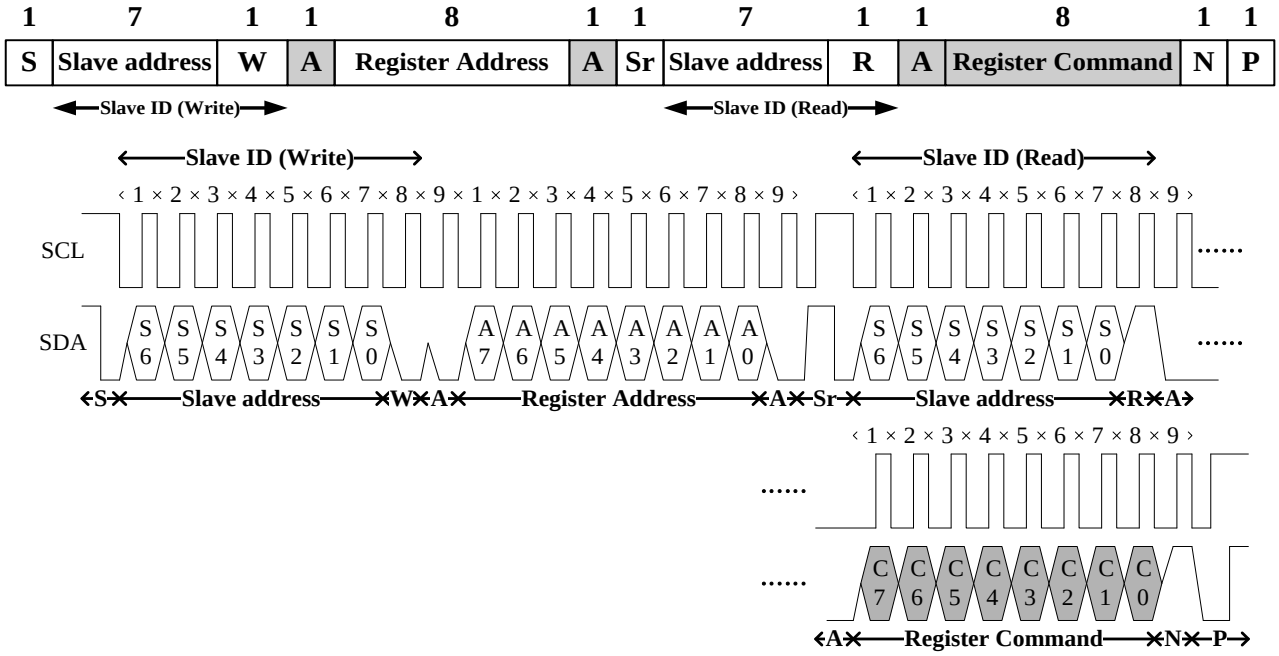


III. I2C Read Protocol





IV. I2C Read (Block Read) Protocol



A	Acknowledge (0 for an ACK)	**N**	Non-Acknowledge(1 for an NACK)
S	Start condition	**Sr**	Repeated Start condition
P	Stop condition		
W	Write (0 for writing)	**R**	Read (1 for read)
■	Slave-to-master	□	Master-to-Slave

V. I2C Slave Address

The device has a 7-bit slave address of 0x53. A read/write bit should be appended to the slave address by the master device to properly communicate with the device.

I^2C Slave Address (Default)									
Command Type	(0x53)							W/R	value
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Write	1	0	1	0	0	1	1	0	0xA6
Read	1	0	1	0	0	1	1	1	0xA7

6. Register Set

Address	R / W	Register Name	Description	Reset Value
0x00	R/W	MAIN_CTRL	ALS/UVS operation mode control, SW reset	0x00
0x04	R/W	ALS_UVS_MEAS_RATE	ALS/UVS measurement rate and resolution in Active Mode	0x22
0x05	R/W	ALS_UVS_GAIN	ALS/UVS analog Gain range	0x01
0x06	R	PART_ID	Part number ID and revision ID	0xB2
0x07	R	MAIN_STATUS	Power-On status, Interrupt status, Data status	0x20
0x0D	R	ALS_DATA_0	ALS ADC measurement data, LSB	0x00
0x0E	R	ALS_DATA_1	ALS ADC measurement data	0x00
0x0F	R	ALS_DATA_2	ALS ADC measurement data, MSB	0x00
0x10	R	UVS_DATA_0	UVS ADC measurement data, LSB	0x00
0x11	R	UVS_DATA_1	UVS ADC measurement data	0x00
0x12	R	UVS_DATA_2	UVS ADC measurement data, MSB	0x00
0x13 – 0x18	R	Reserved	Reserved	0x00
0x19	R/W	INT_CFG	Interrupt configuration	0x10
0x1A	R/W	INT_PST	Interrupt persist setting	0x00
0x21	R/W	ALS_UVS_THRES_UP_0	ALS/UVS interrupt upper threshold, LSB	0xFF
0x22	R/W	ALS_UVS_THRES_UP_1	ALS/UVS interrupt upper threshold, intervening bits	0xFF
0x23	R/W	ALS_UVS_THRES_UP_2	ALS/UVS interrupt upper threshold, MSB	0x0F
0x24	R/W	ALS_UVS_THRES_LOW_0	ALS/UVS interrupt lower threshold, LSB	0x00
0x25	R/W	ALS_UVS_THRES_LOW_1	ALS/UVS interrupt lower threshold, intervening bits	0x00
0x26	R/W	ALS_UVS_THRES_LOW_2	ALS/UVS interrupt lower threshold, MSB	0x00

6.1 MAIN_CTRL Register (Address: 0x00) (Read/Write)

This register controls the operation modes of UVS/ALS, which can be set to either standby or active mode. When writing to this register, it will cause a stop to any ongoing measurements ALS/UVS and start new measurement.

0x00	MAIN_CTRL (default = 0x00)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Reserved			***Software Reset***	***UVS_Mode***	***0***	***ALS/UVS Enable***	***0***

Field	Bits	Default	Description	
Reserved	7:5	000	--	--
SW Reset	4	0	0	Software reset is NOT triggered (default)
			1	Software reset is triggered
UVS_Mode	3	0	0	ALS Mode
			1	UVS Mode
Reserved	2	0	--	Reserved
ALS/UVS Enable	1	0	0	Light sensor (ALS/ or UVS) standby
			1	Light sensor (ALS/ or UVS) active
Reserved	0	0	0	Write as ‘0’

6.2 ALS_UVS_MEAS_RATE Register (Address: 0x04) (Read/Write)

This register controls ALS/UVS measurement resolution, Gain setting and measurement rate. When the measurement rate is programmed to be faster than possible for the programmed ADC measurement, the rate will be lowered than programmed (maximum speed).

0x04	ALS_UVS_MEAS_RATE (default = 0x22)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	0	***ALS/UVS Resolution***			***0***	***ALS/UVS Measurement Rate***		

Field	Bits	Default	Description	
Reserved	7	0	--	--
ALS/UVS Resolution	6:4	010	000	20 Bit, Conversion time = 400ms
			001	19 Bit, Conversion time = 200ms
			010	18 Bit, Conversion time = 100ms(default)
			011	17 Bit, Conversion time = 50ms
			100	16 Bit, Conversion time = 25ms
			101	13 Bit, Conversion time = 12.5ms
			110/111	Reserved
Reserved	3	0	--	Reserved
ALS/UVS Measurement Rate	2:0	010	000	25ms
			001	50ms
			010	100ms (default)
			011	200ms
			100	500ms
			101	1000ms
			110/111	2000ms

6.3 ALS_UVS_GAIN (Address: 0x05) (Read/Write)

This register controls ALS/UVS measurement Gain Range.

0x05	ALS_UVS_GAIN (default = 0x01)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Reserved					***ALS/UVS Gain Range***		

Field	Bits	Default	Description	
Reserved	7:3	00000	--	Reserved
ALS/UVS Gain Range	2:0	001	000	Gain Range: 1
			001	Gain Range: 3 (default)
			010	Gain Range: 6
			011	Gain Range: 9
			100	Gain Range: 18

6.4 PART_ID Register (Address: 0x06) (Read Only)

This register defines the part number and revision identification of the sensor.

0x06	PART_ID (default = 0xB2)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Part Number ID				***Revision ID***			

Field	Bits	Default	Description
Part Number ID	7:4	1011	Part Number ID
Revision ID	3:0	0010	Revision ID of the component. The value increases by one each time a new silicon revision is manufactured.

6.5 MAIN_STATUS Register (Address: 0x07) (Read Only)

This register stores the information about the ALS/UVS interrupts and data status. The interrupt status in Bit 4 determines if the ALS/UVS interrupt criteria are met in Normal Interrupt Mode. It triggers when the UVS/ALS data is above the upper or below the lower threshold for a specified number of consecutive measurements in respective interrupt persist settings.

0x07	MAIN_STATUS (default = 0x20)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Reserved		***Power-On status***	***ALS/UVS Interrupt status***	***ALS/UVS data status***	***Reserved***		

Field	Bits	Default	Description	
Reserved	7:6	00	--	
Power-On Status	5	1	1	Power on event and all interrupt threshold settings in the registers have been reset to power on default states and should be examined if necessary.
			0	This flag is cleared after the register is read.
ALS/UVS Interrupt Status	4	0	0	Interrupt is NOT triggered (default)
			1	Interrupt is triggered and will be cleared after read
UVS/ALS Data Status	3	0	0	UVS/ALS data is old data (Data has been read)
			1	UVS/ALS data is new data (Data has not been read and will be cleared after read)
Reserved	2:0	000	---	

6.6 ALS_DATA Registers (Address: 0x0D/ 0x0E/0x0F) (Read Only)

The Ambient Light Sensor Channel digital output data are expressed as a 13 to 20 bit unsigned integer data. When I^2C read operation is active and points to any of the register address between 0x07 and 0x12, all 3 registers will be locked until the I^2C read operation has been completed or the specified address range is left. This is to ensure that the data in the registers is from the same measurement even if an additional measurement cycle ends during the read operation. New measurement data is stored into temporary registers and the actual ALS_DATA registers will be updated as soon as there is no on-going I^2C read operation to the address range 0x07 to 0x12.

0x0D	ALS_DATA_0 (default = 0x00)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	ALS_DATA_0, Low							

0x0E	ALS_DATA_1 (default = 0x00)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	ALS_DATA_1, Middle							

0x0F	ALS_DATA_2 (default = 0x00)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Reserved				***ALS_DATA_2, High***			

Field	Address	Bits	Default	Description
ALS_Data_0, Low	0x0D	7:0	00000000	ALS Data lower byte data
ALS_Data_1, Middle	0x0E	7:0	00000000	ALS Data Middle byte data
ALS_Data_2, High	0x0F	7:4	0000	Reserved
		3:0	0000	ALS Data Higher byte data

6.7 UVS_DATA Registers (Address: 0x10/0x11/0x12) (Read Only)

The UV Sensor Channel digital output data are expressed as a 16 to 20 bit unsigned integer data. When I^2C read operation is active and points to any of the register address between 0x07 and 0x12, all 3 registers will be locked until the I^2C read operation has been completed or the specified address range is left. This is to ensure that the data in the registers is from the same measurement even if an additional measurement cycle ends during the read operation. New measurement data is stored into temporary registers and the UV Sensor registers will be updated as soon as there is no on-going I^2C read operation to the address range 0x07 to 0x12.

0x10	UVS_DATA_0 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS_Data_0, Low Byte Data							
0x11	UVS_DATA_1 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS_Data_1, Middle Byte Data							
0x12	UVS_DATA_2 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Reserved				*UVS_Data_2, Higher Byte Data*			

Field	Address	Bits	Default	Description
UVS_Data_0	0x10	7:0	00000000	UVS Data lower byte data
UVS_Data_1	0x11	7:0	00000000	UVS Data Middle byte data
UVS_Data_2	0x12	7:4	0000	Reserved
		3:0	0000	UVS Data Higher byte data

6.8 INT_CFG Register (Address: 0x19) (Read/Write)

This register controls the operation of the interrupt pin and functions. The ALS/UVS interrupt is enabled by LS_INT_EN=1 (Bit 2). The ALS/UVS interrupt source generator either uses the ALS_DATA or the UVS_DATA registers as input. The ALS/UVS interrupt source is selected by the LS_INT_SEL bits in the INT_CFG register

0x19	INT_CFG (default = 0x10)							
	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
	Reserved		**LS_INT_SEL**		**Reserved**	**LS_INT_EN**	**Reserved**	

Field	Bits	Default	Description	
Reserved	7:6	00	--	--
LS Interrupt Selection	5:4	01	00	Reserved
			01	ALS Channel (Default)
			10	Reserved
			11	UVS Channel
Reserved	3	0	--	--
LS interrupt enable	2	0	0	LS interrupt disabled (default)
			1	LS interrupt enabled
Reserved	1:0	00	Write as ‘00’	

6.9 INT_PST Register (Address: 0x1A) (Read/Write)

This register INT_PST (Interrupt Persist) sets the ALS/UV persist level. Persist is the N number of times the measurement data is outside the range defined by the upper and lower threshold limits before asserting the interrupt.

0x1A	INT Persist (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ALS/UV_ Persist				*Reserved*			

Field	Bits	Default	Description	
ALS/UV Persist	7:4	0000	0000	Every ALS/UV value out of threshold range asserts an interrupt (default)
			0001	2 consecutive ALS/UV values out of threshold range assert an interrupt
			…	…
			1111	16 consecutive ALS/UV values out of threshold range assert an interrupt
Reserved	3:0	0000	…	

6.10 UVS_ALS_THRES Registers (Address: 0x21/0x22/0x23/0x24/0x25/0x26) (Read/Write)

The UVS/ALS_THRES_UP (up to 20-bits) and UVS/ALS_THRES_LOW (up to 20-bits) registers determines the upper and lower limit of the interrupt threshold value respectively. Interrupt will be triggered if measurement data in DATA_x registers is exceeding the upper and lower limits.

0x21	UVS/ALS_THRES_UP_0 (default = 0xFF)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS/ALS Upper Threshold, Low							
0x22	UVS/ALS_THRES_UP_1 (default = 0xFF)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS/ALS Upper Threshold, Middle							
0x23	UVS/ALS_THRES_UP_2 (default = 0x0F)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Reserved				*UVS/ALS Upper Threshold, Higher*			

0x24	UVS/ALS_THRES_LOW_0 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS/ALS Low Threshold, Low							
0x25	UVS/ALS_THRES_LOW_1 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	UVS/ALS Low Threshold, Middle							
0x26	UVS/ALS_THRES_LOW_2 (default = 0x00)							
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Reserved				*UVS/ALS Low Threshold, Higher*			

Field	Address	Bits	Default	Description
UVS/ALS Upper Threshold, Low	0x21	7:0	11111111	CS/ALS upper interrupt threshold, Low byte
UVS/ALS Upper Threshold, Mid	0x22	7:0	11111111	CS/ALS upper interrupt threshold, Mid byte
UVS/ALS Upper Threshold, Higher	0x23	7:4	0000	Reserved
		3:0	1111	UVS/ALS upper interrupt threshold, Higher byte
UVS/ALS Lower Threshold, Low	0x24	7:0	00000000	UVS/ALS lower interrupt threshold, Low byte
UVS/ALS Lower Threshold, Mid	0x25	7:0	00000000	UVS/ALS lower interrupt threshold, Mid byte
UVS/ALS Lower Threshold, Higher	0x26	7:4	0000	Reserved
		3:0	0000	UVS/ALS lower interrupt threshold, Higher byte

7. ALS/UVI Formula

7.1 ALS Lux Formula

Lux_Calc is the calculated lux reading based on the output ADC from ALS DATA regardless of light sources.

$$Lux_{Calc} = \frac{0.6 \times ALS\_DATA}{(GAIN \times INT)} \times W_{FAC}$$

Where :

1. ALS_DATA = Data stored in the registers (Address: 0x0D-0x0F)
2. For device under tinted window with coated-ink of flat transmission rate at 400-600nm wavelength, window factor is to compensate light loss due to the lower transmission rate from the coated-ink.
   a. WFAC = 1 for NO window / clear window glass.
   b. WFAC >1 device under tinted window glass. Calibrate under white LED.
3. The Gain factors & Integration time factors:

ALS Gain	GAIN
X1	**1**
X3	**3**
X6	**6**
X9	**9**
X18	**18**

Resolution (bit) / Integration Time (ms)	INT
16-bit, 25ms	**0.25**
17-bit, 50ms	**0.5**
18-bit, 100ms	**1**
19-bit, 200ms	**2**
20-bit, 400ms	**4**

7.2 UVI Conversion Formula

$$UVI_{Calc} = \frac{UV\ Sensor\ Count}{UV\ Sensitivity} \times W_{FAC}$$

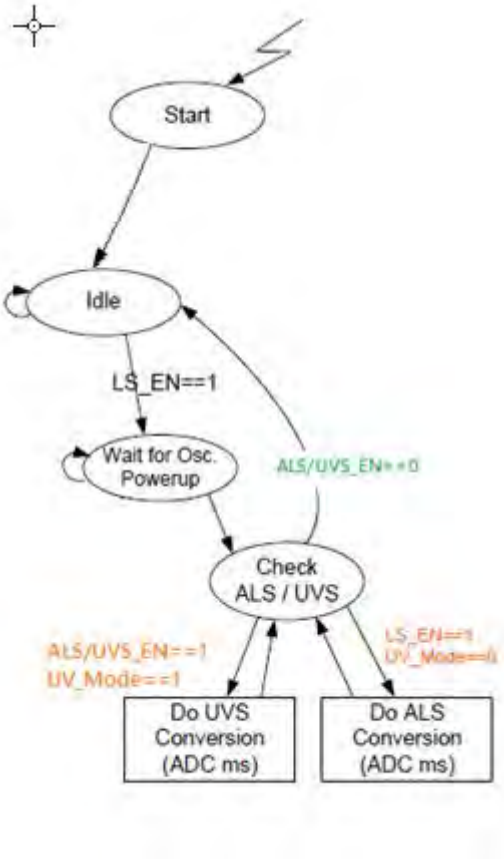
where:

1. WFAC depends on the type of window used. WFAC =1 (no window)

8. Device Operation (State Machine and Interrupt Features)

8.1 State Machine

Below diagram is the main state machine of LTR-390UV.



ALS measurements can be activated by setting the UVS/ALS_Enable bit to 1 and the UVS_Mode bit to 0 in the MAIN_CTRL register. UV measurements can be activated by setting the LS_EN bit to 1 and the UVS_Mode bit to 1 in the MAIN_CTRL register. As soon as ALS or UVS become activated through an I²C™ command, the internal support blocks are powered on. Once the voltages and currents are settled (typically after 5ms), the state machine checks for trigger events from a measurement scheduler to start the ALS or UVS conversions according to the selected measurement repeat rates. Once LS_EN is changed back to 0, a conversion running on the respective channel will be completed and the relevant ADCs and support blocks will move to standby mode.

8.2 Interrupt Features

This device generates Light Sensor (ALS or UVS depending on configuration) interrupt signals and output to the INT pad. The interrupt conditions are always evaluated after completion of a new conversion on the ALS/UV channels.

8.2.1 ALS/UVS Sensor Interrupt

The LS interrupt is enabled by LS_INT_EN=1. The ALS/UVS interrupt source generator either uses the ALS_DATA or the UVS_DATA registers as input. The ALS/UVS interrupt source is selected by the ALS/UVS_INT_SEL bits in the INT_CFG register. The Light Sensor threshold interrupt is enabled with ALS/UVS_INT_EN=1. It is set when the ALS_DATA or UVS_DATA data is above the upper or below the lower threshold for a specified number of consecutive measurements. It is set when the absolute value of the difference between the previous and current ALS/UVS data is above the decoded ALS/UVS variance threshold for a specified number of consecutive measurements (1+ALS/UVS_PERSIST).

9. Pseudo Codes Examples

SLAVE Addresses

```
Slave_Addr = 0x53                    // Slave address
```

MAIN_CTRL Register

// This defines the operating modes of the ALS,UV. Default setting is 0x00 in Standby mode.

```
Register_Addr = 0x00                 //MAIN_CTRL register
Command = 0x02                       // ALS in Active Mode
                                     // UVS in Active Mode, Command = 0x0A
WriteByte(Slave_Addr, Register_Addr, Command);
```

ALS_UVS_MEAS_RATE Register

// This controls the ALS/UVS Resolution & Measurement rate.
// Default setting of the register is 0x22

```
Register_Addr = 0x04                 // ALS_UVS_MEAS_RATE register
Command = 0x22                       // Resolution=18bits, Meas Rate = 100ms
                                     // Resolution=20bits, Meas Rate = 500ms, Command=0x04

WriteByte(Slave_Addr, Register_Addr, Command)
```

ALS_UVS_GAIN Register

// This controls the ALS/UVS GAIN.
// Default setting of the register is 0x01

```
Register_Addr = 0x05                 // ALS_UVS_GAIN register
Command = 0x01                       // Gain Range=3.
                                     // Gain Range=18, Command=0x04
WriteByte(Slave_Addr, Register_Addr, Command)
```

INT_CFG Register

// This controls the interrupt mode of ALS, UVS.
// Default setting of the register is 0x10

```
Register_Addr = 0x19                 // INT_CFG register
Command = 0x14                       // ALS_INT_EN=1.
                                     // UVS_INT_EN=1, Command=0x34
WriteByte(Slave_Addr, Register_Addr, Command)
```

INT_PST Register

// This controls the persistence of interrupt of ALS, UVS.
// Default setting of the register is 0x00

```
Register_Addr = 0x1A                 // INT_CFG register
Command = 0x00                       // ALS/UVS Persist=0.
                                     // ALS/UVS Persist=1, Command=0x10

WriteByte(Slave_Addr, Register_Addr, Command)
```

ALS_DATA Registers (Read Only)

//The registers 0x0D, 0x0E & 0x0F contain ALS data, up to 20bits.

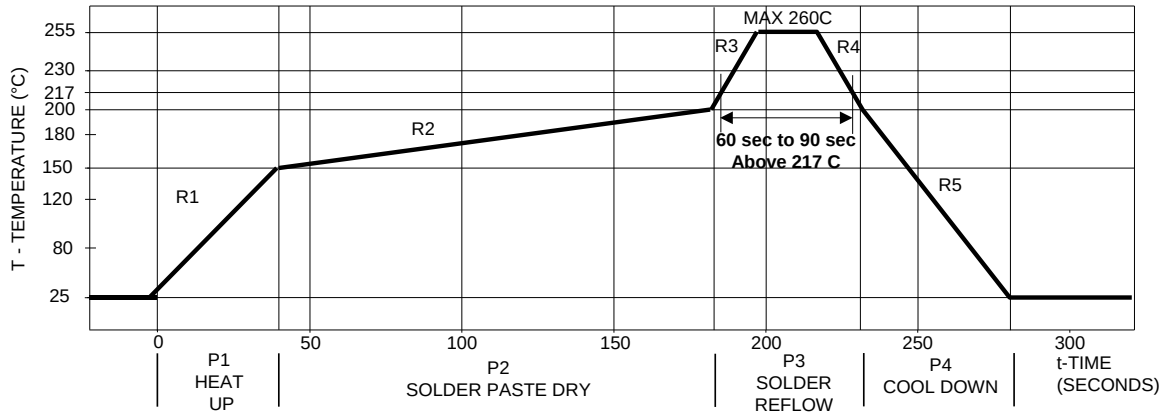
```
Register_Addr = 0x0D                          // ALS_DATA_0 address
Data1 = ReadByte(Slave_Addr, Register_Addr)
Register_Addr = 0x0E                          // ALS_DATA_1 address
Data2 = ReadByte(Slave_Addr, Register_Addr)
Register_Addr = 0x0F                          // ALS_DATA_2 address
Data3 = ReadByte(Slave_Addr, Register_Addr)   // ALS_DATA=Data3*65536+Data2*256+Data1.
```

UVS_DATA Registers (Read Only)

//The registers 0x10, 0x11 & 0x12 contain UVS data, up to 20bits.

```
Register_Addr = 0x10                          // UVS_DATA_0 address
Data1 = ReadByte(Slave_Addr, Register_Addr)
Register_Addr = 0x11                          // UVS_DATA_1 address
Data2 = ReadByte(Slave_Addr, Register_Addr)
Register_Addr = 0x12                          // UVS_DATA_2 address
Data3 = ReadByte(Slave_Addr, Register_Addr)   // UVS_DATA=Data3*65536+Data2*256+Data1.
```

10. Recommended Leadfree Reflow Profile



Process Zone	Symbol	ΔT	Maximum ΔT/Δtime or Duration
Heat Up	P1, R1	25°C to 150°C	3°C/s
Solder Paste Dry	P2, R2	150°C to 200°C	100s to 180s
Solder Reflow	P3, R3 P3, R4	200°C to 260°C 260°C to 200°C	3°C/s -6°C/s
Cool Down	P4, R5	200°C to 25°C	-6°C/s
Time maintained above liquid's point , 217°C		> 217°C	60s to 90s
Peak Temperature		260°C	-
Time within 5°C of actual Peak Temperature		> 255°C	20s
Time 25°C to Peak Temperature		25°C to 260°C	8mins

It is recommended to perform reflow soldering no more than twice.

11. Moisture Proof Packaging

All LTR-390UV-01 are shipped in moisture proof package. Once opened, moisture absorption begins. This part is compliant to JEDEC J-STD-033A Level 3.

Time from Unsealing to Soldering

After removal from the moisture barrier bag, the parts should be stored at the recommended storage conditions and soldered within seven days. When the moisture barrier bag is opened and the parts are exposed to the recommended storage conditions for more than seven days, the parts must be baked before reflow to prevent damage to the parts.

1. Recommended Storage Conditions

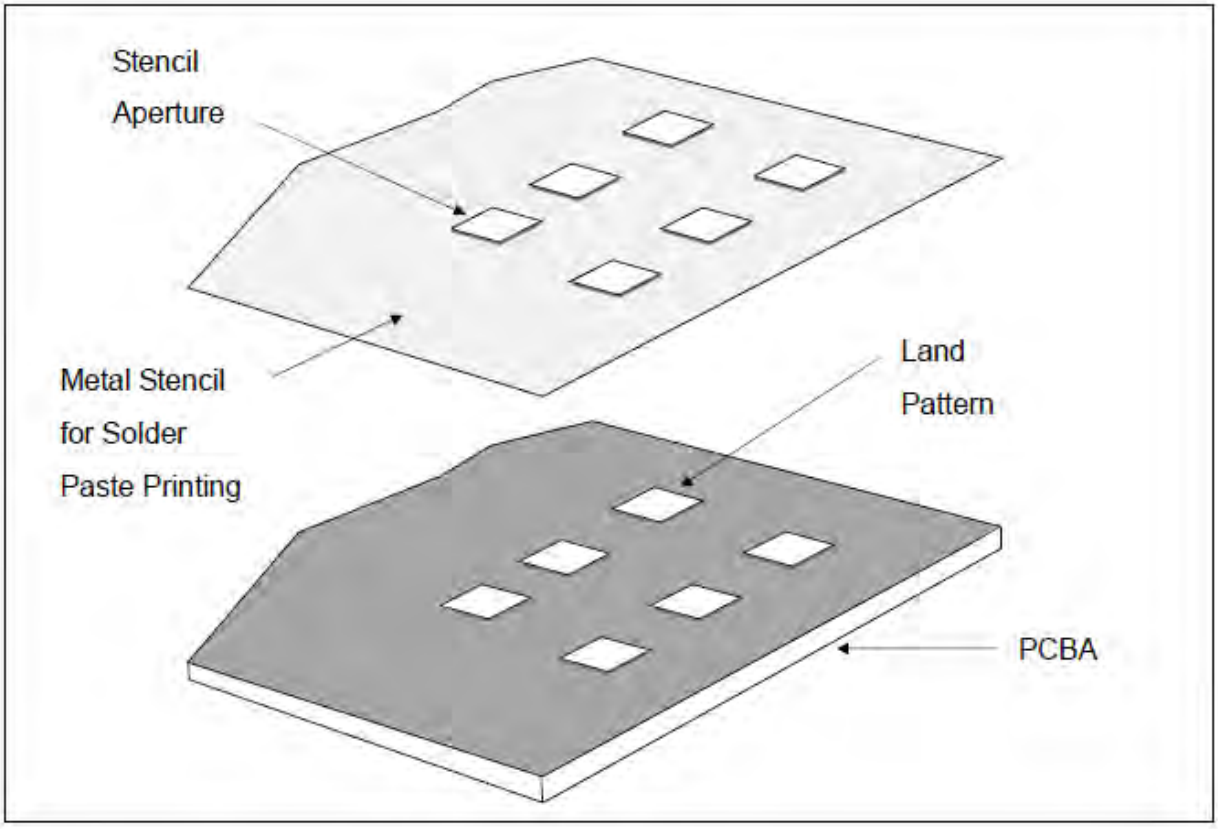
Storage Temperature	10°C to 30°C
Relative Humidity	Below 60% RH

2. Baking Conditions

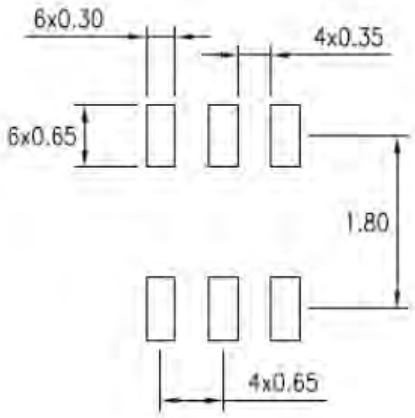
Package	Temperature	Time
In Reels	60°C	48 hours
In Bulk	100°C	4 hours

Baking should only be done once.

12. Recommended Land Pattern



I. Recommended Land Pattern for LTR-390UV-01

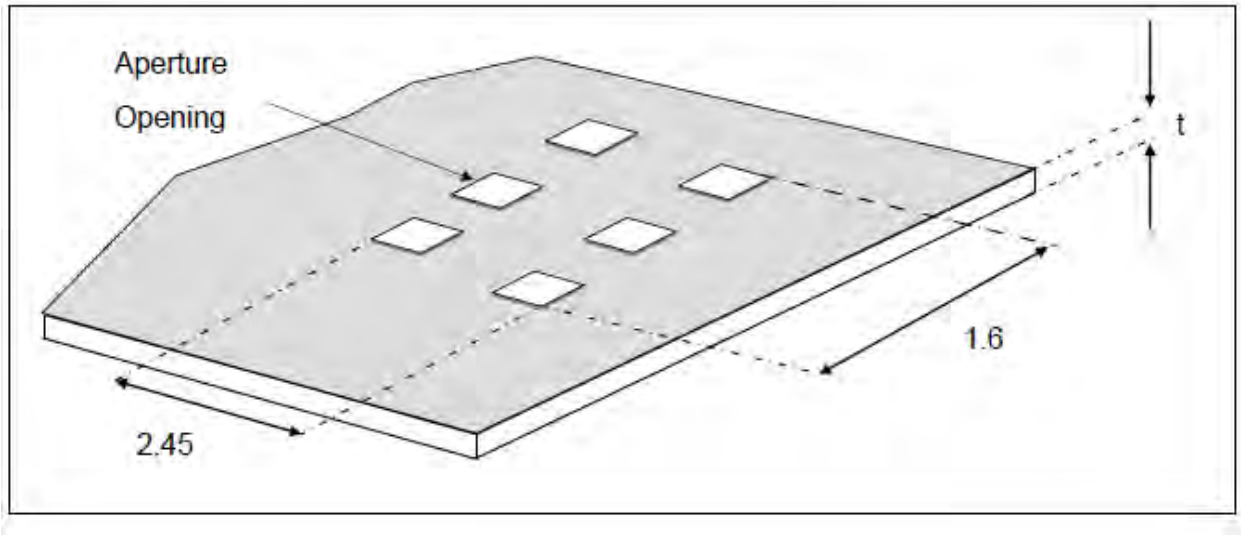


Note: All dimensions are in millimeters

13. Metal Stencil Aperture

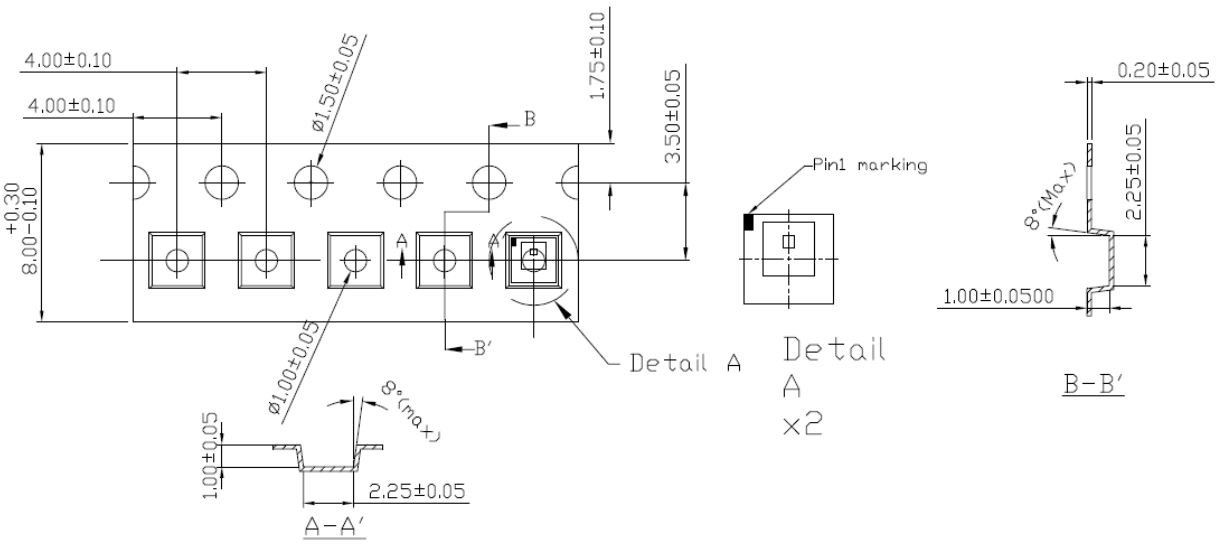
It is recommended that the metal stencil used for solder paste printing has a thickness (t) of 0.11mm (0.004 inches / 4 mils) or 0.127mm (0.005 inches / 5 mils).

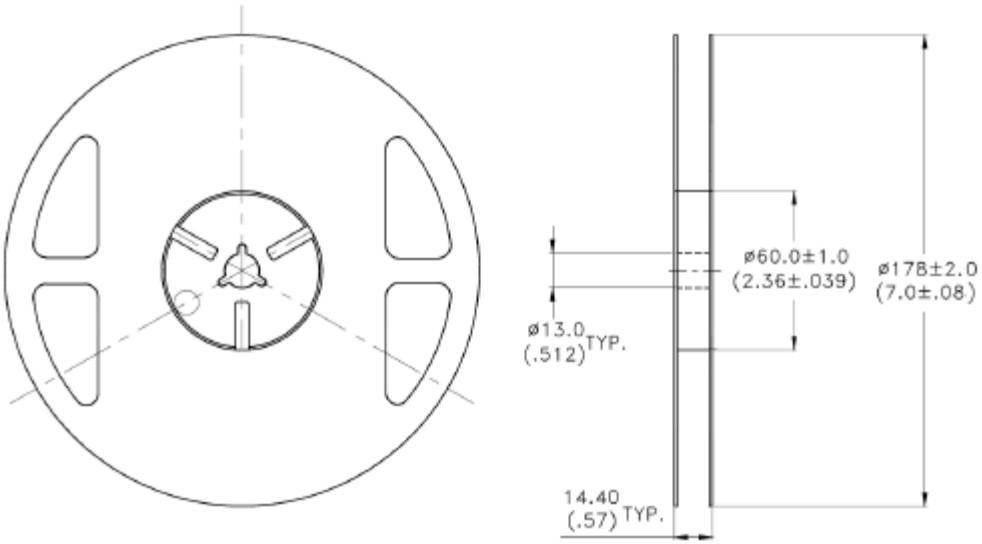
The stencil aperture opening is recommended to be 0.3mm x 0.65mm which has the same dimension as the land pattern. This is to ensure adequate printed solder paste volume and yet no shorting.



Note:
1. All dimensions are in millimeters

14. Tape and Reel Dimensions





Notes:

1. All dimensions are in millimeters (inches)
2. Empty component pockets sealed with top cover tape
3. 7 inch reel - 2500 pieces per reel
4. In accordance with ANSI/EIA 481-1-A-1994 specifications

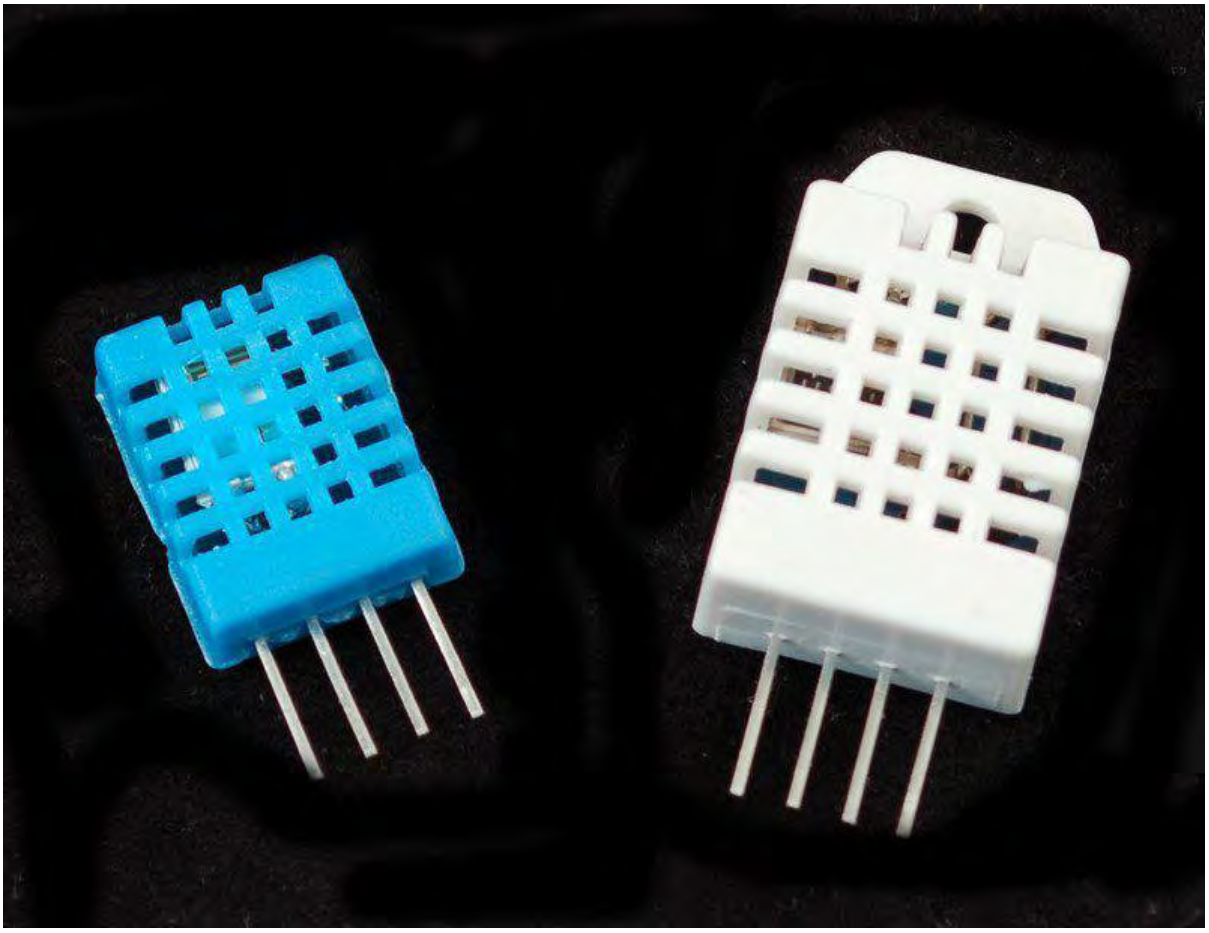
Revision Table:

Version	Update	Page	Date
1.0	Datasheet as created	Total 30	28-Jul-15
1.1	ALS lux formula updated	Total 33	24-Aug-15

B.8 Datasheet: DHT22

DHT11, DHT22 and AM2302 Sensors

Created by lady ada



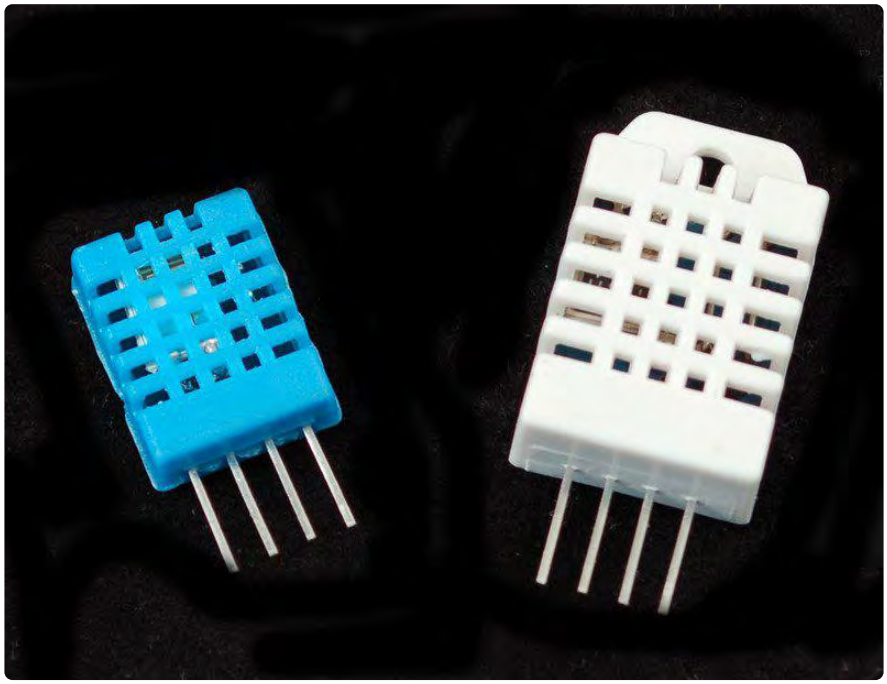
https://learn.adafruit.com/dht

Last updated on 2021-12-13 12:38:56 PM EST

Table of Contents

Overview

This tutorial covers the low cost DHT temperature & humidity sensors (https://adafru.it/aJU). These sensors are very basic and slow, but are great for hobbyists who want to do some basic data logging. The DHT sensors are made of two parts, a capacitive humidity sensor and a thermistor (https://adafru.it/aHD). There is also a very basic chip inside that does some analog to digital conversion and spits out a digital signal with the temperature and humidity. The digital signal is fairly easy to read using any microcontroller.



DHT11 vs DHT22

We have two versions of the DHT sensor, they look a bit similar and have the same pinout, but have different characteristics. Here are the specs:

DHT11 (http://adafru.it/386)

- Ultra low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

DHT22 (http://adafru.it/385)/ AM2302 (https://adafru.it/uF2) (Wired version)

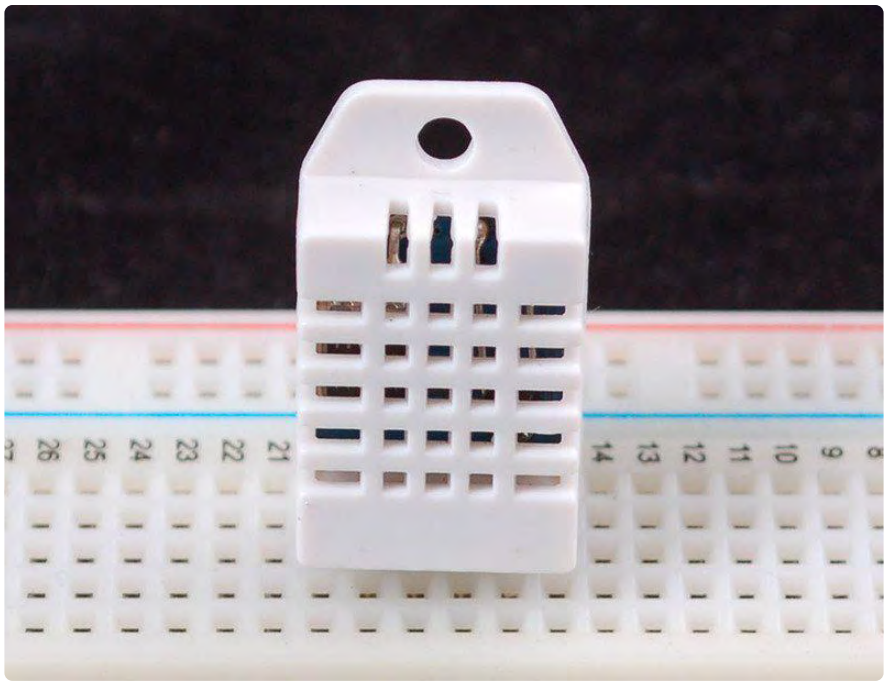
- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 80°C temperature readings ±0.5°C accuracy
- No more than 0.5 Hz sampling rate (once every 2 seconds)
- Body size 15.1mm x 25mm x 7.7mm
- 4 pins with 0.1" spacing

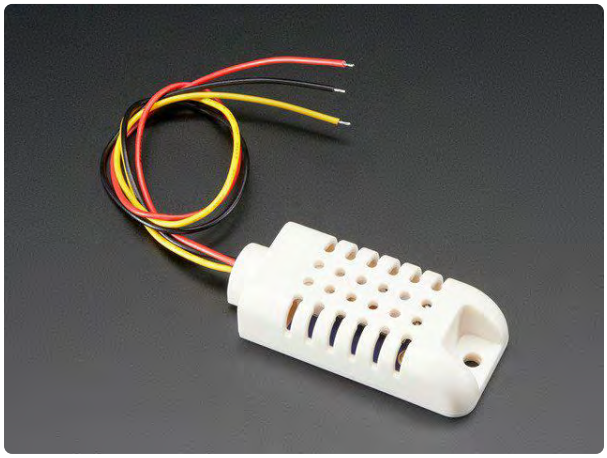
As you can see, the DHT22 (http://adafru.it/385) / AM2302 (https://adafru.it/uF2) is a little more accurate and good over a slightly larger range. Both use a single digital pin and are 'sluggish' in that you can't query them more than once every second or two.

You can pick up both the DHT11 (http://adafru.it/386) and DHT22 (http://adafru.it/385) or AM2302 (https://adafru.it/uF2) from the adafruit shop!

Connecting to a DHTxx Sensor

Luckily it is trivial to connect to these sensors, they have fairly long 0.1"-pitch pins so you can plug them into any breadboard, perfboard or similar.





AM2302 (wired DHT22) temperature-humidity sensor
The AM2302 is a wired version of the DHT22, in a large plastic body. It is a basic, low-cost digital temperature and humidity sensor....
https://www.adafruit.com/product/393

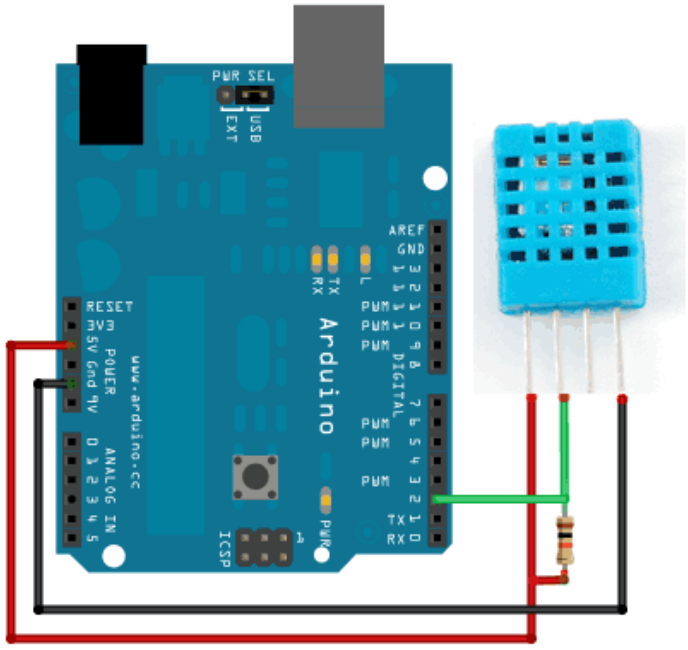
Likewise, it is fairly easy to connect up to the DHT sensors. They have four pins

1. VCC - red wire Connect to 3.3 - 5V power. Sometime 3.3V power isn't enough in which case try 5V power.
2. Data out - white or yellow wire
3. Not connected
4. Ground - black wire

Simply ignore pin 3, its not used. You will want to place a 10 Kohm resistor between VCC and the data pin, to act as a medium-strength pull up on the data line. The Arduino has built in pullups you can turn on but they're very weak, about 20-50K

DHT22 and AM2302 often have a pullup already inside, but it doesn't hurt to add another one!

This diagram shows how we will connect for the testing sketch. Connect data to pin 2, you can change it later to any pin.



If you have an AM2302

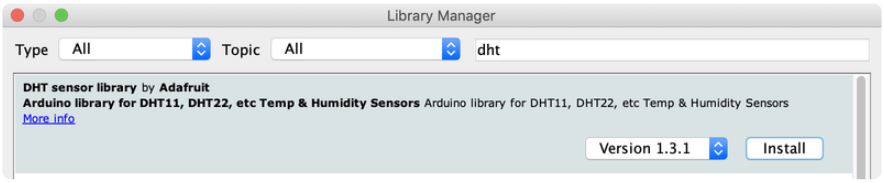
Using a DHTxx Sensor

To test the sketch, we'll use an Arduino. You can use any micrcontroller that can do microsecond timing, but since its a little tricky to code it up, we suggest verifying the wiring and sensor work with an Arduino to start.

You should have the Arduino IDE (https://adafru.it/fvm) software running at this time. Next it's necessary to install our DHT library, which can be done though the Arduino Library Manager:

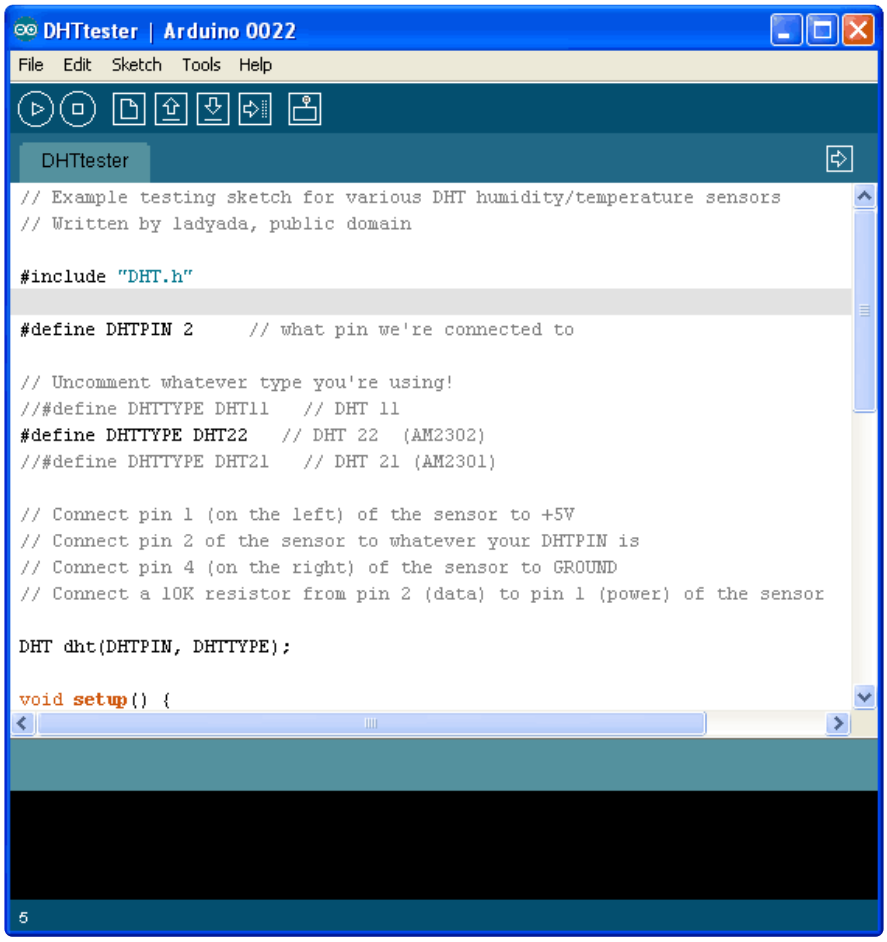
Sketch→Include Library→Manage Libraries...

Enter "dht" in the search field and look through the list for "DHT sensor library by Ad afruit." Click the "Install" button, or "Update" from an earlier version.



IMPORTANT: As of version 1.3.0 of the DHT library you will also need to install the Adafruit Unified Sensor library, which is also available in the Arduino Library Manager:

Now load up the Examples→DHT→DHTtester sketch



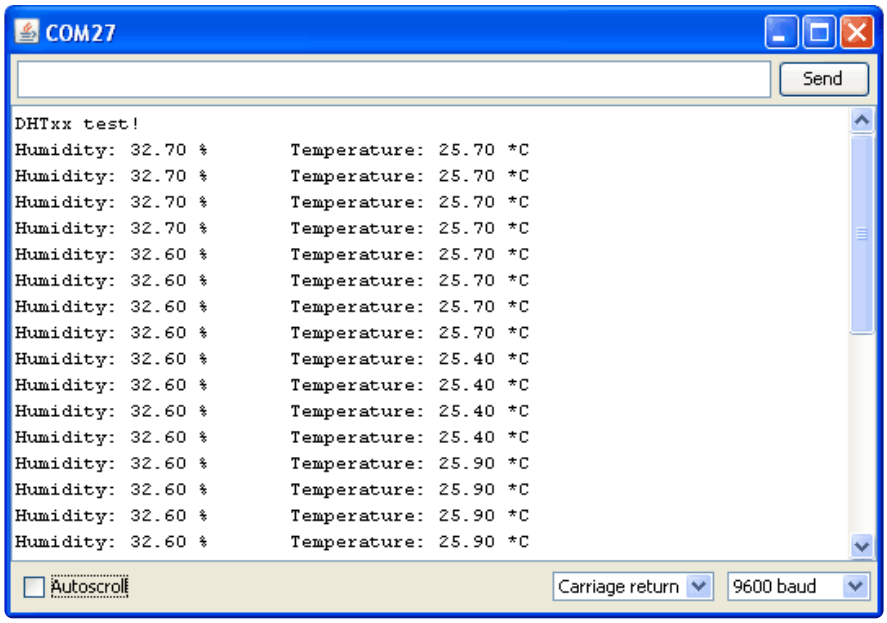
If you're using a DHT11 sensor, comment out the line that sets the type:

```
//#define DHTTYPE DHT22   // DHT 22  (AM2302)
```

and uncomment the line that says:

```
#define DHTTYPE DHT11   // DHT 11
```

This will make the data appear correctly for the correct sensor. Upload the sketch!



You should see the temperature and humidity. You can see changes by breathing onto the sensor (like you would to fog up a window) which should increase the humidity.

You can add as many DHT sensors as you line on individual pins, just add new lines such as

```
DHT dht2 = DHT(pin, type);
```

below the declaration for the initial `dht` object, and you can reference the new `dht2` whenever you like.

DHT CircuitPython Code

This library uses the pulseio module in CircuitPython. As of CircuitPython 7.0.0, pulseio is no longer available on the smallest CircuitPython builds, such as the Trinket M0, Gemma M0, and Feather M0 Basic boards. You can substitute a more modern sensor, which will work better as well. See the guide Modern Replacements for DHT11 and DHT22 Sensors (https://learn.adafruit.com/modern-replacements-for-dht11-dht22-sensors) for suggestions.

Adafruit CircuitPython Module Install

To use the DHT sensor with your Adafruit CircuitPython board you'll need to install the Adafruit_CircuitPython_DHT (https://adafru.it/Beq) module on your board.

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/Em8) for your board.

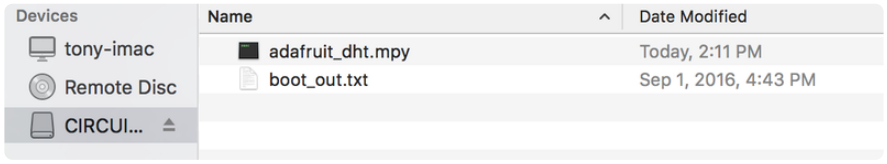
Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). Our introduction guide has a great page on how to install the library bundle (https://adafru.it/ABU) for both express and non-express boards.

Remember for non-express boards like the, you'll need to manually install the necessary libraries from the bundle:

- adafruit_dht.mpy

You can also download the adafruit_dht.mpy from its releases page on Github (https://adafru.it/Ber).

Before continuing make sure your board's lib folder or root filesystem has the adafruit_dht.mpy module copied over.

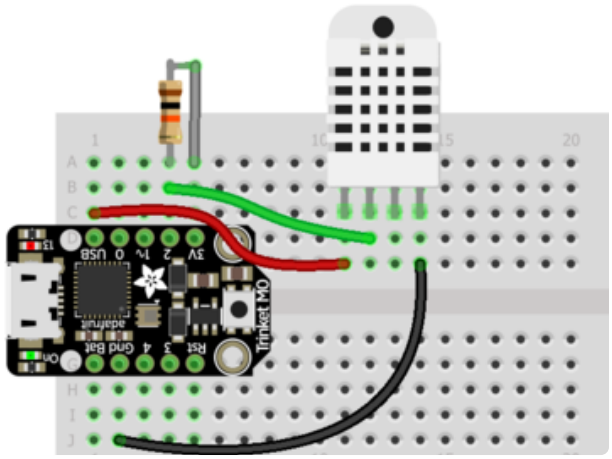


Wiring

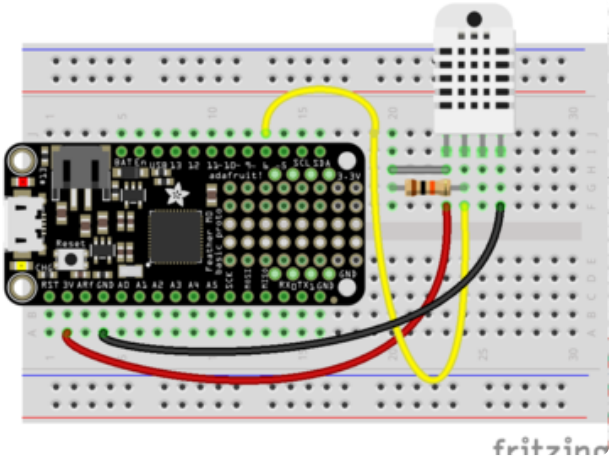
DHT wiring is very simple:

- The left-most pin is power. We recommend powering from 5V (sometimes 3V is not enough) - this is OK even if you are using 3.3V logic
- The second pin is data. Connect a 10K pullup resistor from this pin to 3.3V. If you are using a DHT11 it's required. If you're using a DHT22 or AM2302 you can sometimes leave this off
- Skip the third pin
- The right-most pin is ground

For the DATA pin you must pick a pin that has PWM support (pulseio) - Check the board's guide for what pins have timers available



Here's an example using a Trinket M0 - you can use any CircuitPython board, just check that the Data pin is `pulseio`-capable.



In this example we'll use a Feather M0 and DHT22 sensor connected to pin D6

Fritzing Source

https://adafru.it/A0o

Usage

To demonstrate the usage of the DHT sensor module you can connect to your board's serial REPL and run Python code to read the temperature and humidity.

Next connect to the board's serial REPL (https://adafru.it/Awz)so you are at the CircuitPython >>> prompt.

Next import the board and adafruit_dht modules, these are necessary modules to initialize and access the sensor:

```
import board
import adafruit_dht
```

You may also want to try powering the DHT sensor from 5V (we found sometimes it really needs more power) but still having the 10K pull-up resistor to 3.3V volts)

Now create an instance of either the DHT11 or DHT22 class, depending on the type of sensor you're using (for the AM2302 sensor use the DHT22 class). You must pass in the pin which is connected to the signal line, for example a DHT22 or AM2302 sensor connected to board pin `D6` would need this code:

```
dht = adafruit_dht.DHT22(board.D6)
```

Note for a DHT11 sensor you'd instead use adafruit_dht.DHT11 in place of the adafruit _dht.DHT22 code above.

At this point you're all set and ready to start reading the temperature and humidity! You can do this by reading the temperature property which returns temperature in degrees Celsius:

```
dht.temperature
```

```
>>> dht.temperature
22.2
>>>
```

To read the humidity grab the value of the humidity property, it will return the percent humidity as a floating point value from 0 to 100%:

```
dht.humidity
```

```
>>> dht.humidity
53.2
>>>
```

In most cases you'll always get back a temperature or humidity value when requested, but sometimes if there's electrical noise or the signal was interrupted in some way you might see an exception thrown to try again. It's normal for these sensors to sometimes be hard to read and you might need to make your code retry a few times if it fails to read. However if you always get errors and can't ever read the sensor then double check your wiring (don't forget the pull-up resistor if needed!) and the power to the device.

Example Code

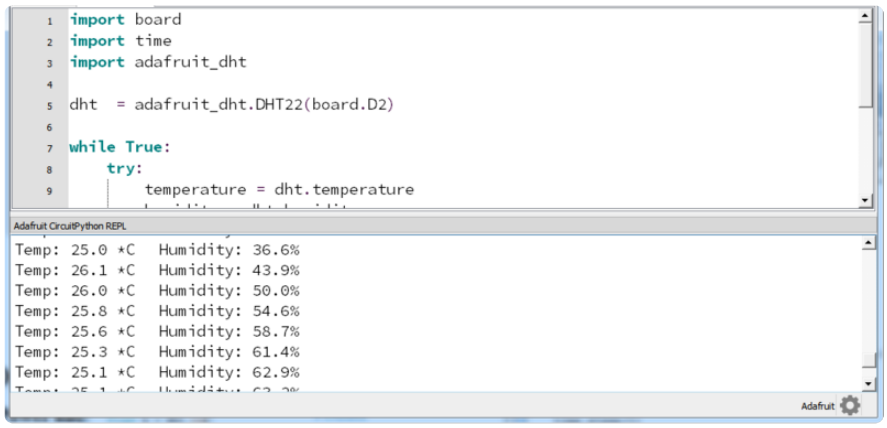
Here's a full example sketch which also manages error-retry logic (which will happen once in a while.

Don't forget to change the logic pin to whatever pin you're using! Then save this as `main.py` on your CircuitPython board

```
import time

import adafruit_dht
import board

dht = adafruit_dht.DHT22(board.D2)

while True:
    try:
        temperature = dht.temperature
        humidity = dht.humidity
        # Print what we got to the REPL
        print("Temp: {:.1f} *C \t Humidity: {}%".format(temperature, humidity))
    except RuntimeError as e:
        # Reading doesn't always work! Just print error and we'll try again
        print("Reading from DHT failure: ", e.args)

    time.sleep(1)
```

If you are using a DHT11, change the code to use a `adafruit_dht.DHT11(board.D2)` object.

Open the REPL to see the output! Breathe on the sensor to see it move temperature and humidity up (unless you are a White Walker in which case the temperature will go down)



Python Docs

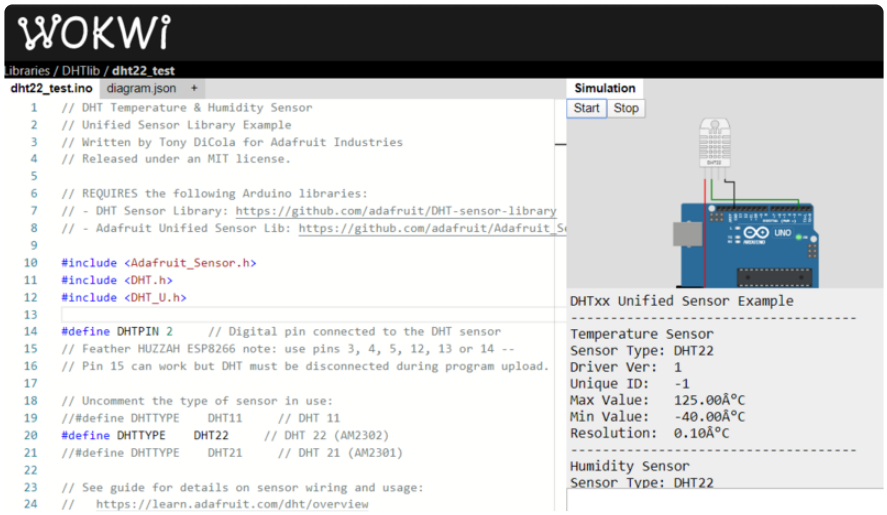
Python Docs (https://adafru.it/C4a)

Downloads

- Arduino library and example code for DHT sensors (https://adafru.it/aJX)
- Adafruit_Sensor library (https://adafru.it/aZm) (required by the DHT library above)
- DHT11 datasheet (https://adafru.it/aJY)(in chinese, so see the DHT22 datasheet too!)
- DHT22 datasheet (https://adafru.it/aJZ)
- K&R Smith calibration notes (https://adafru.it/BfU)

Simulator

You can try out a DHT simulator by Wowki (https://adafru.it/N8B) here: https://wokwi.com/arduino/libraries/DHT-sensor-library (https://adafru.it/Ncg)



Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Adafruit:
393 385 386

References

[1] GAOHOU. (n.d.). PH0-14 Value Detect Sensor Module + PH Electrode Probe BNC For Arduino [Online]. Available: https://www.amazon.com/dp/B0799BXMVJ?psc=1&ref=ppx_yo2ov_dt_b_product_details

[2] IPC, "IPC-2221C - Generic Standard on Printed Board Design," IPC International, Bannockburn, IL, Standard, Dec. 2023. [Online]. Available: https://shop.ipc.org/2221-STD-0-D-0-EN-C

[3] Boveda. "Boveda Size 60 (4 Pack) 69% RH for Cigar." Boveda, [online] Available: https://store.bovedainc.com/collections/for-cigars/products/boveda-size-60-4-pack-69-rh-for-cigar

[4] T. Tonygo2, "LTR390 Micropython Code," Pimoroni Community Forum, 04-Dec-2021. [Online]. Available: https://forums.pimoroni.com/t/ltr390-micropython-code/22314/2.

[5] "CrocSee DC 12V Mini Food Grade Self Priming Diaphragm Fresh Water Transfer Pump 1.3LPM, Replacement Pump for Ice Maker, Coffee Machine, Water Dispenser," Amazon. [Online]. Available: https://www.amazon.com/CrocSee-Diaphragm-Transfer-Replacement-Dispenser/dp/B09XH1GYYQ.

[6] "Gikfun 12V DC Dosing Pump Peristaltic Dosing Head with Connector for Arduino Aquarium Lab Analytic DIY AE1207," Gikfun Store. [Online]. Available: https://www.amazon.com/Gikfun-Peristaltic-Connector-Aquarium-Analytic/dp/B01IUVHB8E.

[7] "Plant Grow LED Light, OUEVA 16.4ft/5M 5050 SMD Waterproof Full Spectrum Red Blue 5:1 Growing Lamp for Aquarium Greenhouse Hydroponic Plant, Garden Flowers Veg Grow Light," OUEVA Store, [Online]. Available: https://www.amazon.com/Plant-OUEVA-16-4ft-Waterproof-Spectrum/dp/B06XCJG4M6.

[8] "USB Black Light Strip, 6.6ft 10W UV LED Blacklight String Lights, 395-400nm, DC 5V, 120 Lamp Beads, Glow in the Dark for Halloween, Birthday, Party, Fluorescent Poster, Room, Bedroom Decoration," GREENIC Store, [Online]. Available: https://www.amazon.com/Blacklight-395-400nm-Halloween-Fluorescent-Decoration/dp/B0B99L22B8.

[9] Lite-On Inc., "LTR-390UV Final Datasheet." [Online]. Available: https://optoelectronics.liteon.com/upload/download/DS86-2015-0004/LTR-390UV_Final_%20DS_V1%201.pdf.

[10] Infineon Technologies AG, "IRLZ34NPBF Datasheet." [Online]. Available: https://www.infineon.com/dgdl/irlz34npbf.pdf?fileId=5546d462533600a40153567206892720.

[11] Rubycon Corporation, "ZLH Series Aluminum Electrolytic Capacitors." [Online]. Available: https://www.rubycon.co.jp/wp-content/uploads/catalog-aluminum/ZLH.pdf.

[12] ON Semiconductor, "1N4001, 1N4002, 1N4003, 1N4004, 1N4005, 1N4006, 1N4007 Axial-Lead Glass Passivated Standard Recovery Rectifiers Datasheet." [Online]. Available: https://www.onsemi.com/pdf/datasheet/1n4001-d.pdf.

[13] "PH Sensor PH-4502C Datasheet." [Online]. Available: https://cdn.awsli.com.br/969/969921/arquivos/ph-sensor-ph-4502c.pdf.

[14] Raspberry Pi Foundation, "Raspberry Pi Pico W Datasheet." [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf.

[15] Texas Instruments, "SN54S151 Data Selector/Multiplexer Datasheet." [Online]. Available: https://www.ti.com/lit/ds/symlink/sn54s151.pdf?ts=1714290231547.