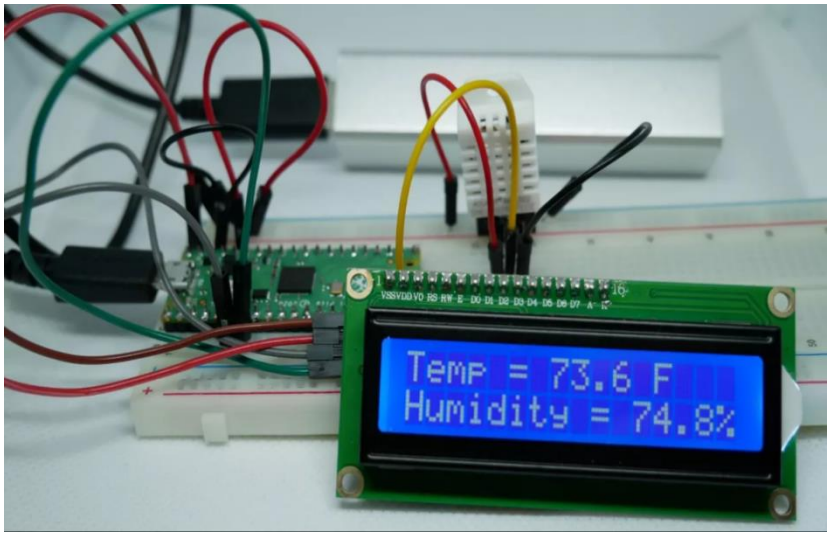
VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY

(Department of Electronics and Communication Engineering)

EMBEDDED AND REAL TIME SYSTEMS (EC8791)

Mini Project Report



PROJECT TITLE : WEATHER STATION

TEAM MEMBERS:

1. Afrin Jumana M (19ECE64)
2. Jeeva Getzie Cynthia A (19ECE72)
3. Srinithi A (19ECE93)
4. Priya M (19ECE83)

EVALUATION TABLE:

Project Summary	10	
Components	10	
Objectives	40	
Outcomes	20	
Specification Detail	20	
Total	100	

PROJECT SUMMARY

INTRODUCTION

Weather forecasts are very important for the functioning of our day-to-day lives. They also play a crucial role in agriculture, cattle farming, forestry, hydroelectric dams, wildfire monitoring, disaster management, and numerous other sectors. In the absence of effective weather monitoring systems, natural disasters, and ecological changes cannot be forecasted. It can result in economic loss and fatalities. The regional weather forecasts are generic in nature. They do not provide ample information about weather conditions in a specific area. The solution to this problem is the development of a Raspberry Pi based Weather Station.

Many things affect the weather. And weather also have effect on most of living as well as non-living things. At weather station study of different environmental parameters using some instruments and equipment has been done. Apart from government and non-government organizations the weather forecasted data can also be used for the fields like agriculture, transportation, construction etc. Apart for the scientific and commercial applications, weather forecasting systems can be used for educational purposes. The data of the measured parameters are not useful if they are not transmitted fast and accurate manner to the users. Therefore, transmitted and processing the measured data is a very important aspect of the modern weather forecast.

In this project, we made Mini Weather Station using DHT11 Humidity Temperature Sensor & Raspberry Pi Pico. A weather station is a system of integrated components that automatically measure, record, and sometimes transmit weather data.

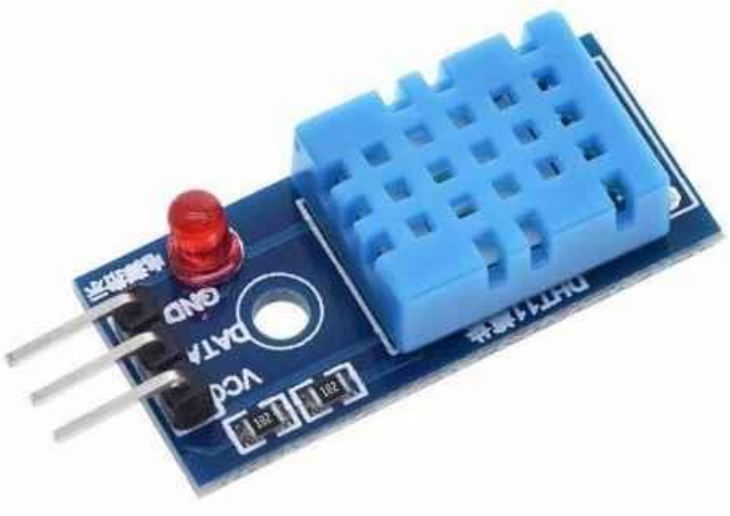
In this project, the ambient temperature value and humidity value are monitored in real-time through the DHT11 Temperature & Humidity sensor and displayed on the 16×2 I2C LCD module synchronously. When the temperature is too high or the humidity is too low, it will trigger the LED light to turn on and off to remind.

COMPONENTS REQUIRED

- Raspberry Pi Pico
- 16X 2 Serial LCD Module Display
- DHT11 Temperature and Humidity Sensor
- Jumper Wire
- Breadboard

DHT11 Temperature & Humidity Sensor:

DHT11 Temperature & Humidity sensor is a temperature and humidity composite sensor with calibrated digital signal output, it includes a resistive humidity sensing element and an NTC temperature measuring element, and is connected to a high-performance 8-bit microcontroller.



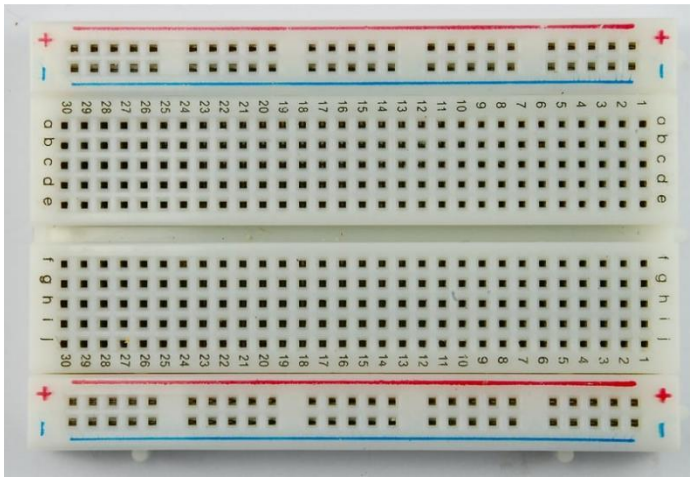
Its application-specific digital module acquisition technology and temperature and humidity sensing technology ensure that the product has extremely high reliability and excellent long-term stability.

Raspberry Pi Pico:

The Raspberry Pi Pico is a tiny, yet powerful microcontroller that was released in January 2021. It is based on the **RP2040** chip and is designed to be used with the Raspberry Pi. The Pico is a great microcontroller for beginners and professional engineers alike. It is very affordable and straightforward to use, yet still has a lot of features that more developed users will appreciate.

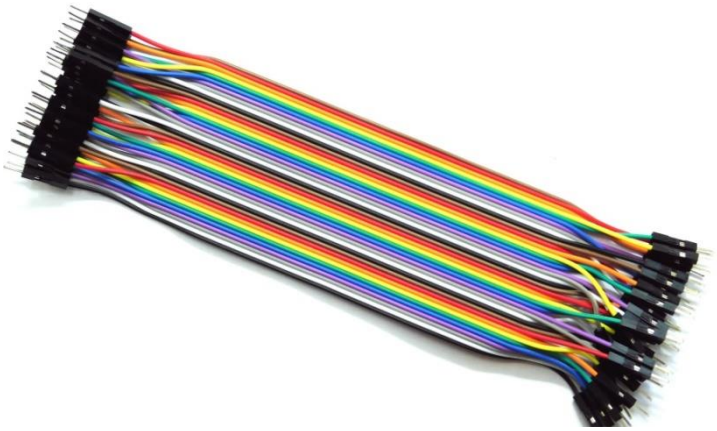
Breadboard:

A breadboard is a construction base for prototyping of electronics. These solderless breadboards does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. A modern solderless breadboard socket consists of a perforated block of plastic with numerous tin plated phosphor bronze or nickel silver alloy spring clips under the perforations. Interconnecting wires and the leads of discrete components such as capacitors, resistors, and inductors, power supply, one or more signal generators, LED display or LCD modules, and logic probes can be inserted into the remaining free holes to complete the circuit. A bus strip usually contains two rows: one for ground and one for a supply voltage. Typically the row intended for a supply voltage is marked in red, while the row for ground is marked in blue or black.



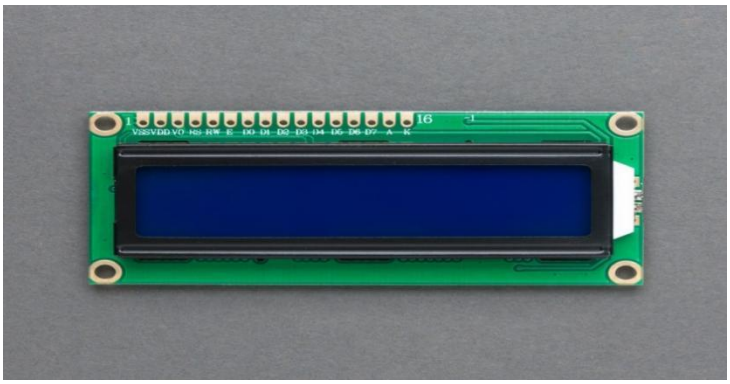
Jumper wires:

Jump wires (also called jumper wires) for solderless bread boarding can be obtained in ready- to-use jump wire sets or can be manually manufactured. The latter can become tedious work for larger circuits. Ready-to-use jump wires come in different qualities, some even with tiny plugs attached to the wire ends. Jump wire material for ready-made should usually be solid copper, tin-plated wire - assuming no tiny plugs are to be attached to the wire ends. Shorter stripped wires might result in bad contact with the board's spring clips (insulation being caught in the springs). Longer stripped wires increase the likelihood of short-circuits on the board. Needle-nose pliers and tweezers are helpful when inserting or removing wires, particularly on crowded boards.

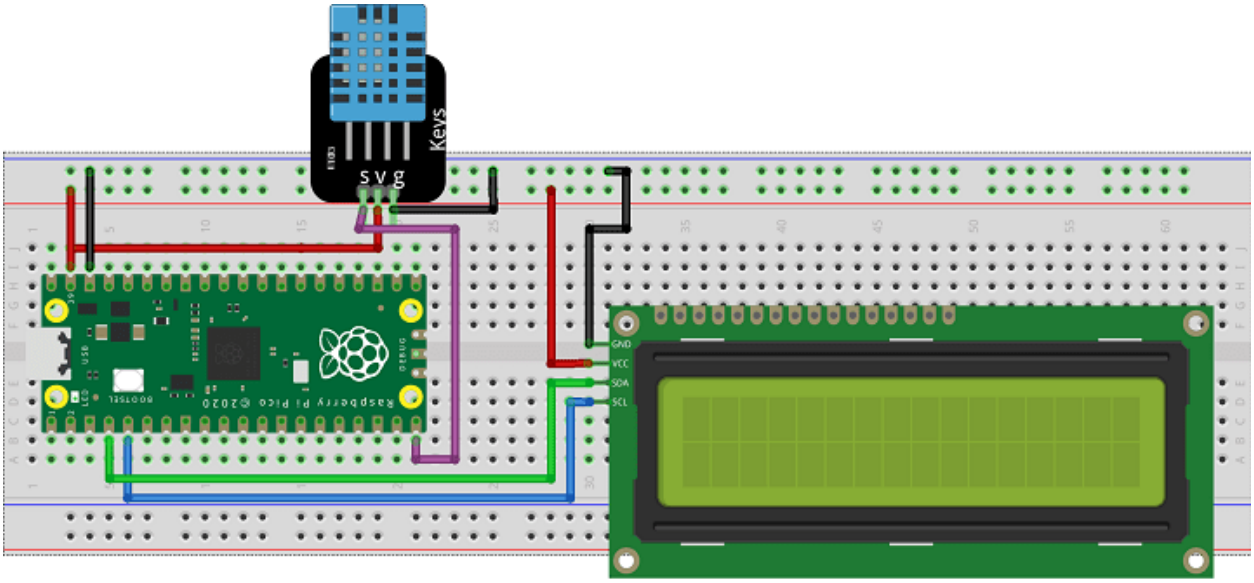


16*2 LCD Module Display:

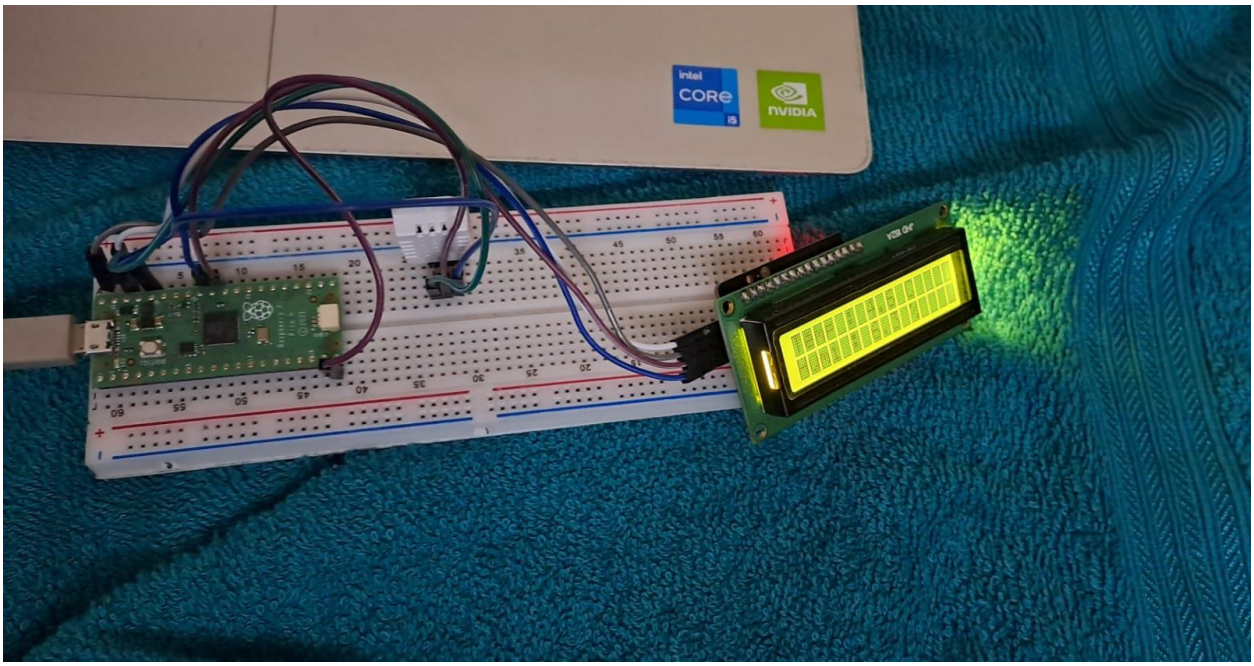
An LCD (Liquid Crystal Display) screen is an electronic display module and has a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.



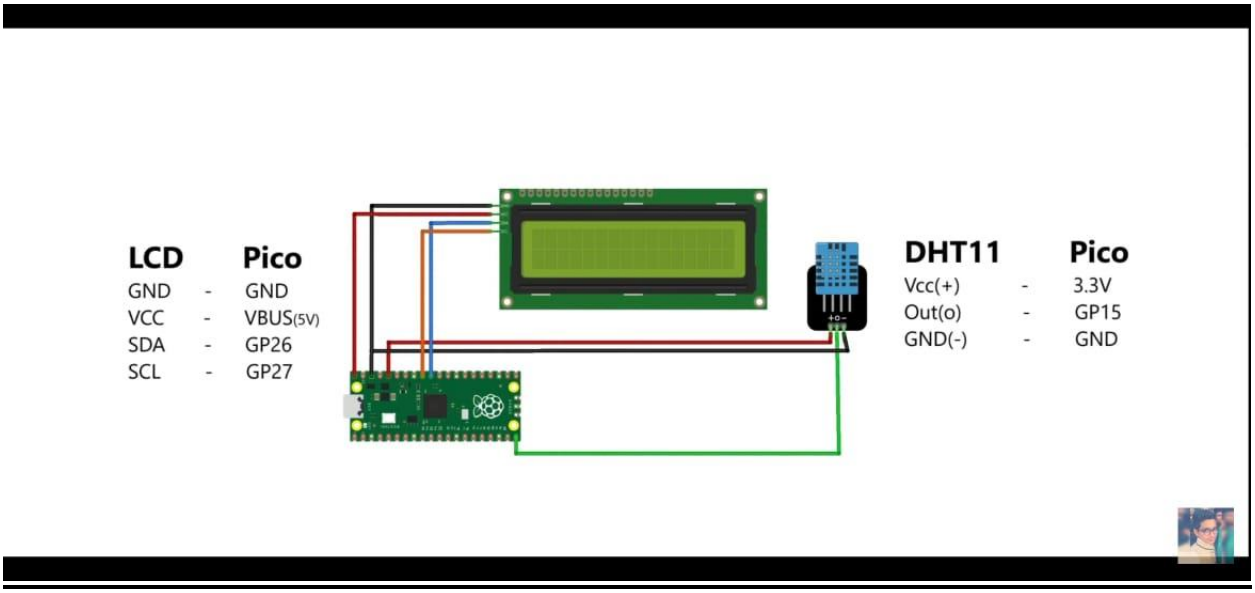
CIRCUIT DIAGRAM:



EXPERIMENTAL SETUP:



PIN CONNECTIONS:



OBJECTIVE:

This project aims to show the current Humidity, Temperature values on the LCD using Raspberry Pi, which makes it a Raspberry Pi Weather Station. You can install this setup anywhere and can monitor the weather conditions of that place from anywhere in the world over the internet, it will show the current data .

Weather prediction is a very important factor, which forecasts the climate in a region based upon the values of weather parameters. So the calculated results from this system can be made use in forecasting the weather of that locality for a period of time.

The project can be enhanced by using a sensor to note the soil moisture value such that usage of unnecessary fertilizers can be reduced. A water meter can be added to estimate the amount of water used water irrigation and thus giving cost estimation. Further, it also reduces the investment of farmers.

The technology changes day by day. Here in this, we make use of raspberry pi pico. In the future there will be more advanced hardware on which we can implement this weather monitoring system. By including the sensors of soil moisture, PH values, and other we can use this in agricultural fields. So, that it would be helpful to farmers to take care of crop yield.

We can also implement an app which supports the android and other operating systems. So, that we can check the data from anywhere at any time by using the internet. It is very easy to install the app and check the data whenever we want. This will be more beneficial for everyone as in every home there is at least one smart phone in these days.

This mini weather station can be made much more compact and reliable with the inclusion of miniature components and by increasing the scaling factor. Also, it is very economical so that with low cost we can take the readings more accurate.

SPECIFICATIONS :

Raspberry Pi Pico specifications:

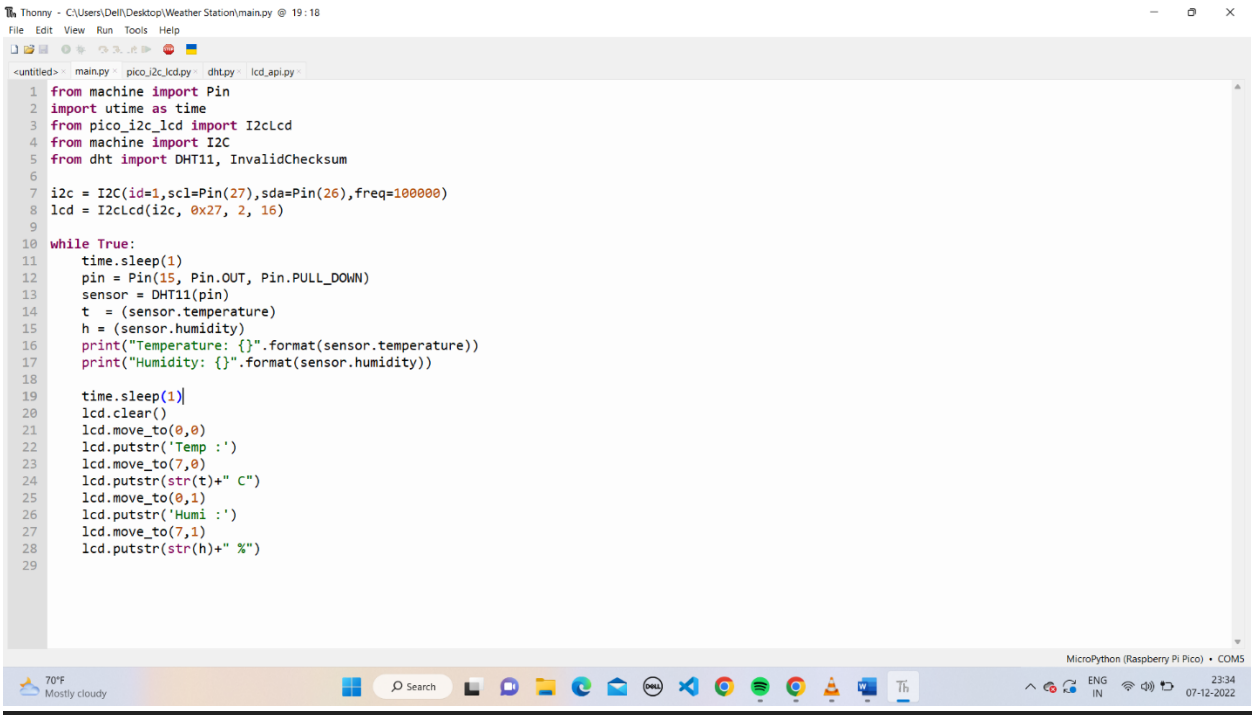
- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core ARM Cortex-M0+ processor, flexible clock running up to 133MHz
- 264kB of SRAM, and 2MB of on-board flash storage
- Castellated module allows soldering direct to carrier boards
- USB 1.1 Host and Device support
- Low-power sleep and dormant modes
- Drag & drop programming using mass storage over USB
- 26 multifunction GPIO pins
- 2× SPI, 2× I2C, 2× UART, 3× 12-bit ADC, 16× controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Fast floating-point libraries in ROM
- 8× Programmable IO (PIO) state machines for custom peripheral support

DHT11 Specifications:

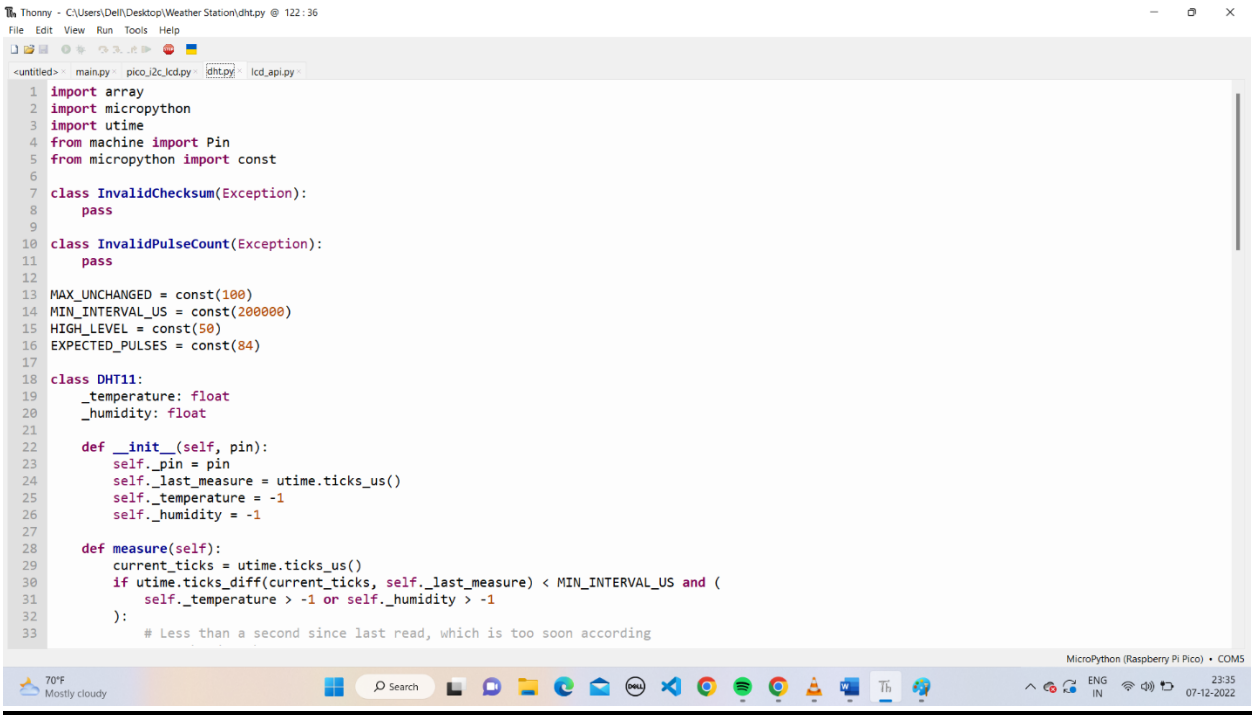
- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

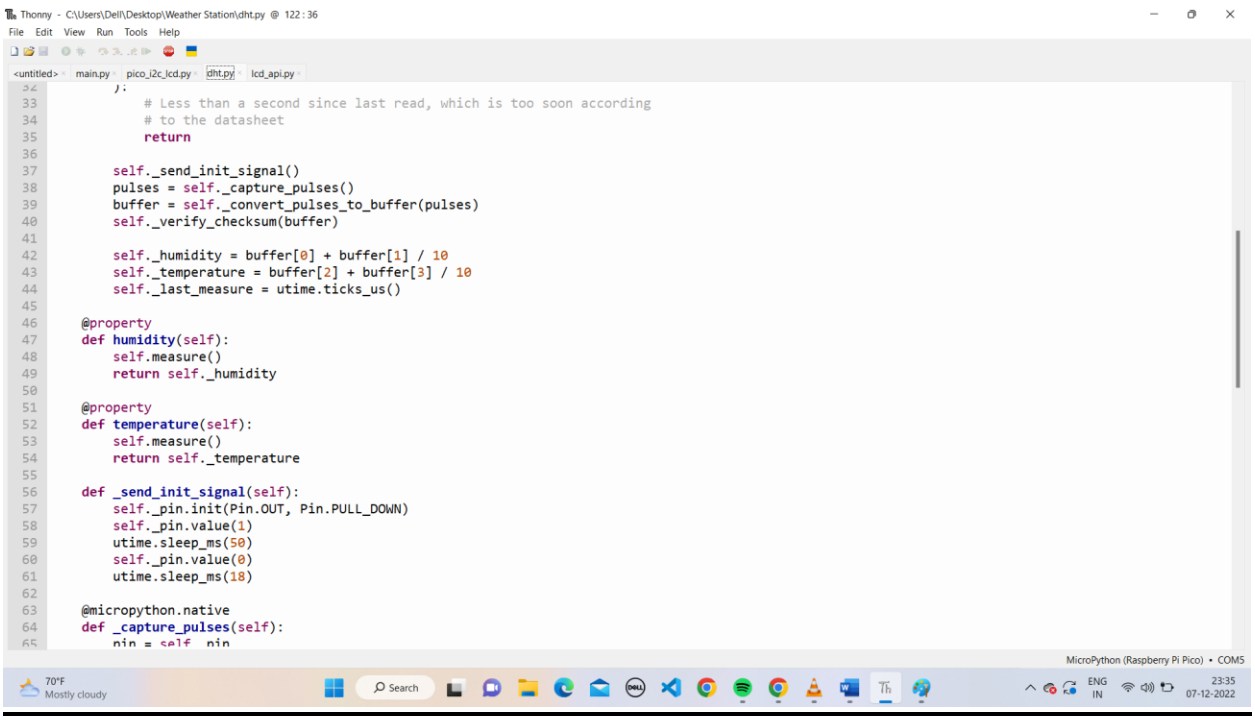
CODE:

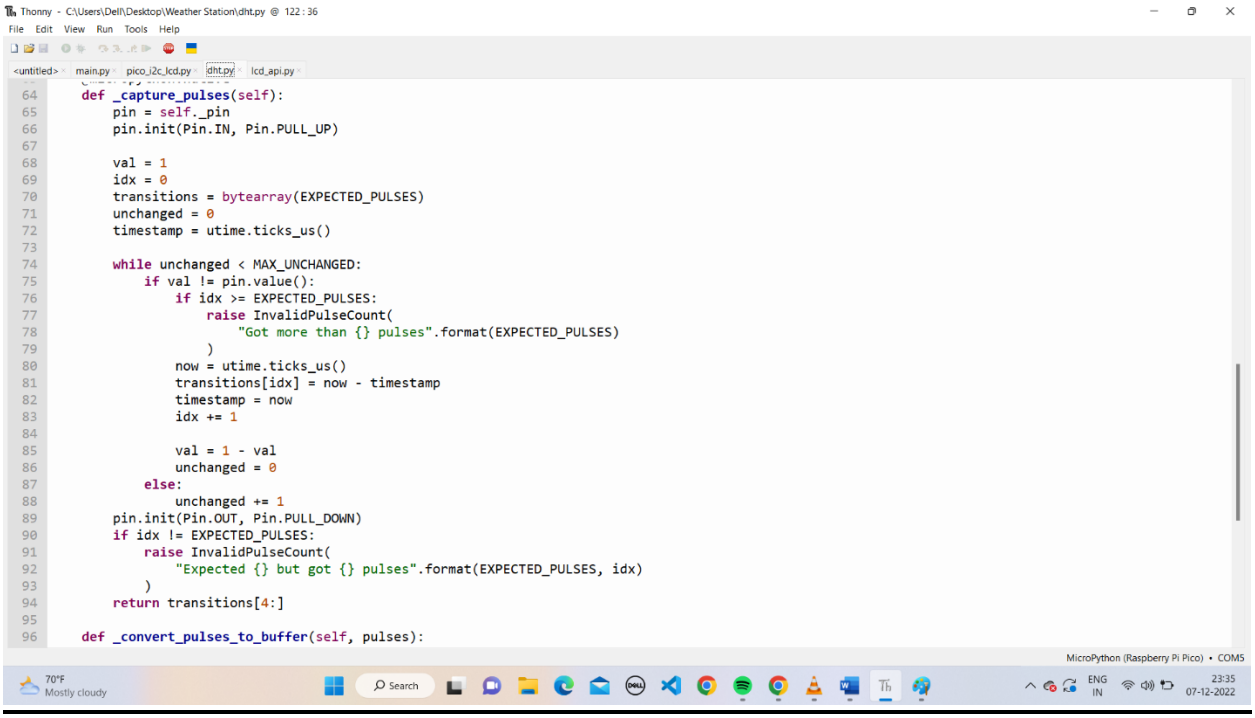
MAIN PROGRAM:

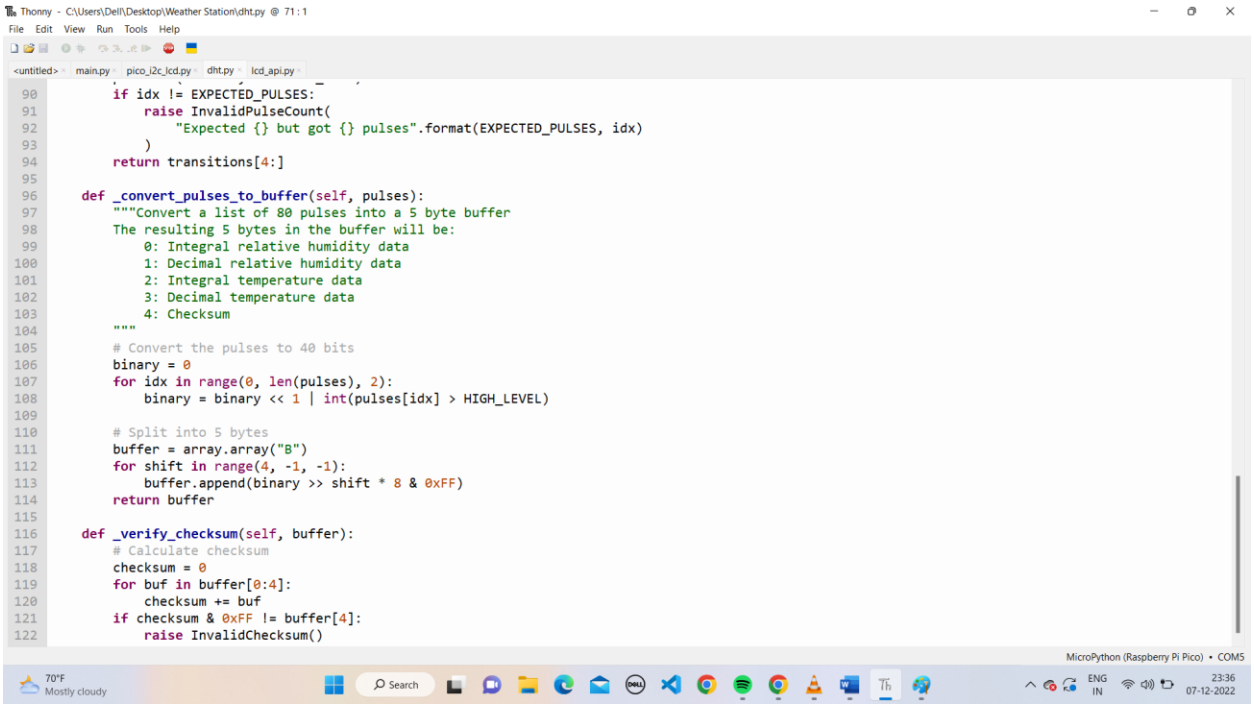


DHT11 :

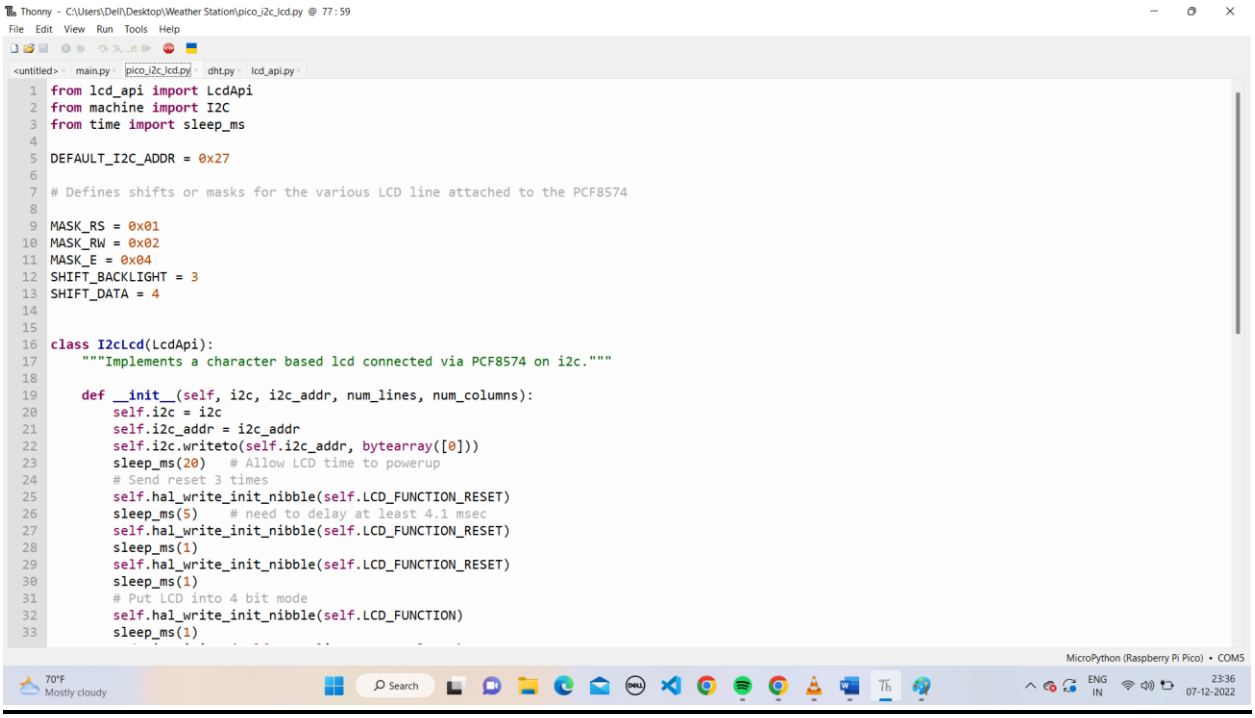


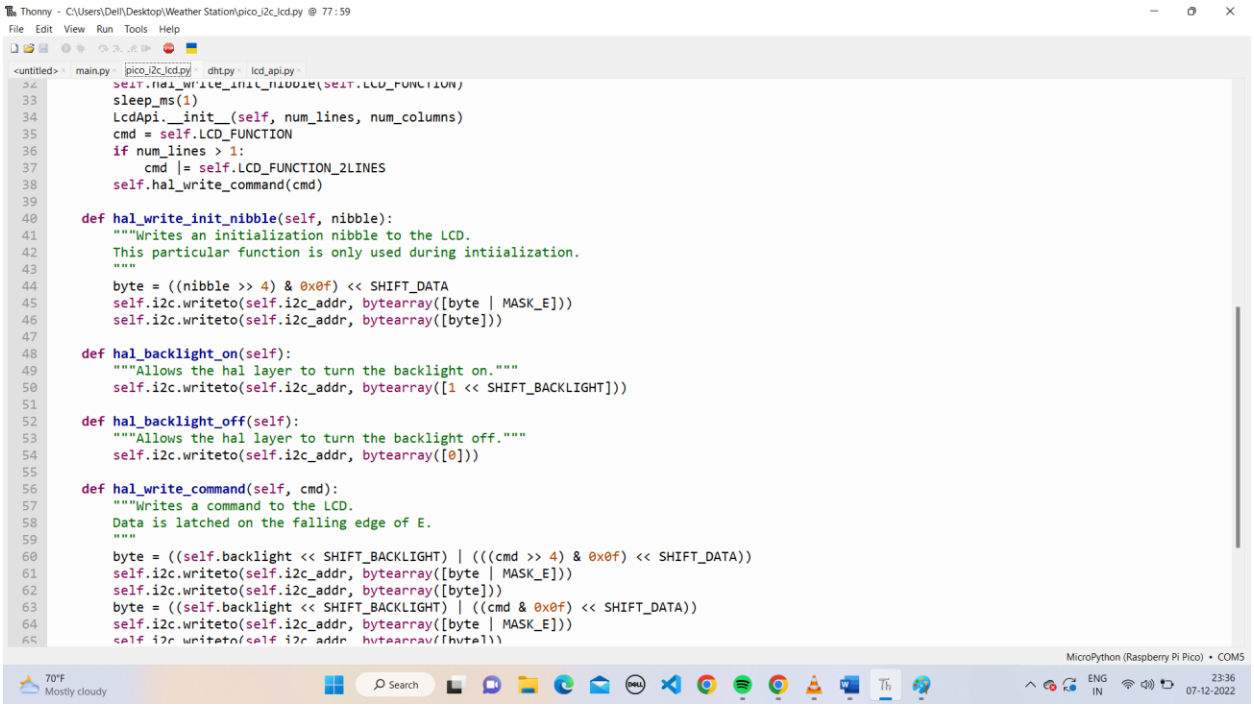

Thonny - C:\Users\Dell\Desktop\Weather Station\dht.py @ 122 : 36
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
):
Less than a second since last read, which is too soon according
to the datasheet
return
self._send_init_signal()
pulses = self._capture_pulses()
buffer = self._convert_pulses_to_buffer(pulses)
self._verify_checksum(buffer)
self._humidity = buffer[0] + buffer[1] / 10
self._temperature = buffer[2] + buffer[3] / 10
self._last_measure = utime.ticks_us()
@property
def humidity(self):
self.measure()
return self._humidity
@property
def temperature(self):
self.measure()
return self._temperature
def _send_init_signal(self):
self._pin.init(Pin.OUT, Pin.PULL_DOWN)
self._pin.value(1)
utime.sleep_ms(50)
self._pin.value(0)
utime.sleep_ms(18)
@micropython.native
def _capture_pulses(self):
MicroPython (Raspberry Pi Pico) • COM5
70°F
Mostly cloudy
Search
ENG
IN
23:35
07-12-2022

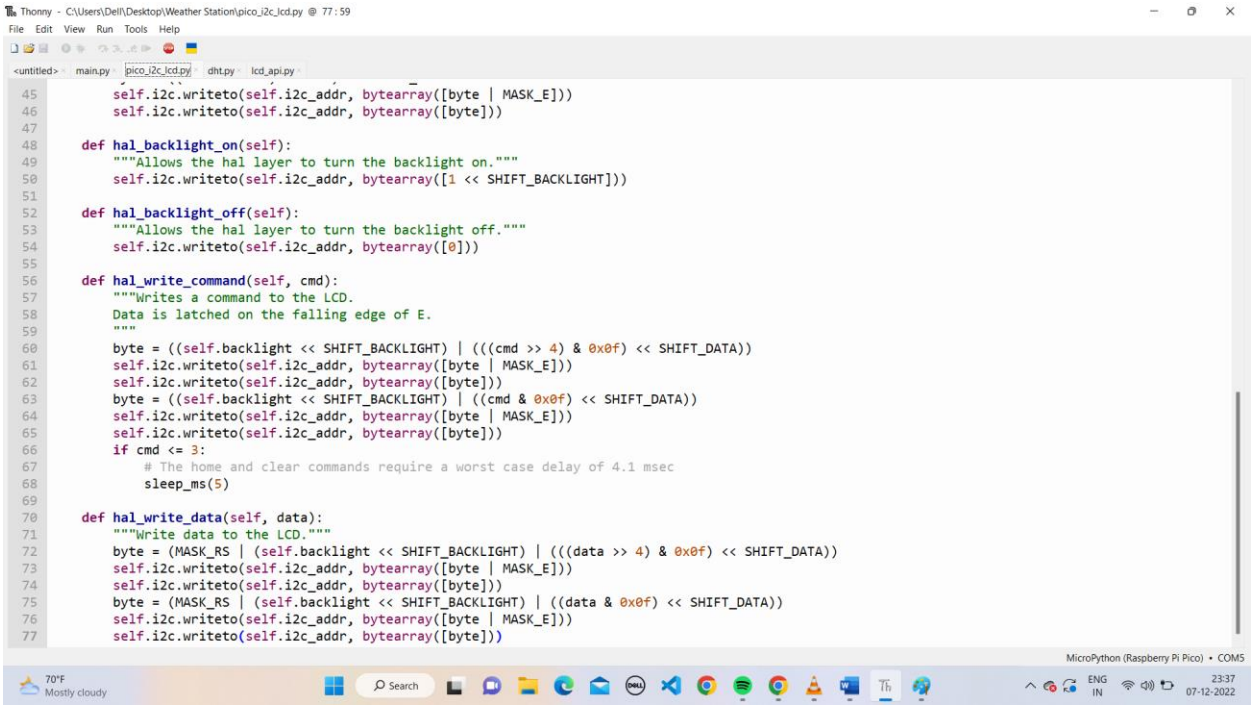

Thonny - C:\Users\Dell\Desktop\Weather Station\dht.py @ 122 : 36
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
def _capture_pulses(self):
pin = self._pin
pin.init(Pin.IN, Pin.PULL_UP)
val = 1
idx = 0
transitions = bytearray(EXPECTED_PULSES)
unchanged = 0
timestamp = utime.ticks_us()
while unchanged < MAX_UNCHANGED:
if val != pin.value():
if idx >= EXPECTED_PULSES:
raise InvalidPulseCount(
"Got more than {} pulses".format(EXPECTED_PULSES)
)
now = utime.ticks_us()
transitions[idx] = now - timestamp
timestamp = now
idx += 1
val = 1 - val
unchanged = 0
else:
unchanged += 1
pin.init(Pin.OUT, Pin.PULL_DOWN)
if idx != EXPECTED_PULSES:
raise InvalidPulseCount(
"Expected {} but got {} pulses".format(EXPECTED_PULSES, idx)
)
return transitions[4:]
def _convert_pulses_to_buffer(self, pulses):
MicroPython (Raspberry Pi Pico) • COM5
70°F
Mostly cloudy
Search
ENG
IN
23:35
07-12-2022


Thonny - C:\Users\Dell\Desktop\Weather Station\dht.py @ 71 : 1
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
if idx != EXPECTED_PULSES:
raise InvalidPulseCount(
"Expected {} but got {} pulses".format(EXPECTED_PULSES, idx)
)
return transitions[4:]
def _convert_pulses_to_buffer(self, pulses):
"""Convert a list of 80 pulses into a 5 byte buffer
The resulting 5 bytes in the buffer will be:
0: Integral relative humidity data
1: Decimal relative humidity data
2: Integral temperature data
3: Decimal temperature data
4: Checksum
"""
# Convert the pulses to 40 bits
binary = 0
for idx in range(0, len(pulses), 2):
binary = binary << 1 | int(pulses[idx] > HIGH_LEVEL)
# Split into 5 bytes
buffer = array.array("B")
for shift in range(4, -1, -1):
buffer.append(binary >> shift * 8 & 0xFF)
return buffer
def _verify_checksum(self, buffer):
# Calculate checksum
checksum = 0
for buf in buffer[0:4]:
checksum += buf
if checksum & 0xFF != buffer[4]:
raise InvalidChecksum()
MicroPython (Raspberry Pi Pico) • COM5
70°F
Mostly cloudy
Search
ENG
IN
23:36
07-12-2022

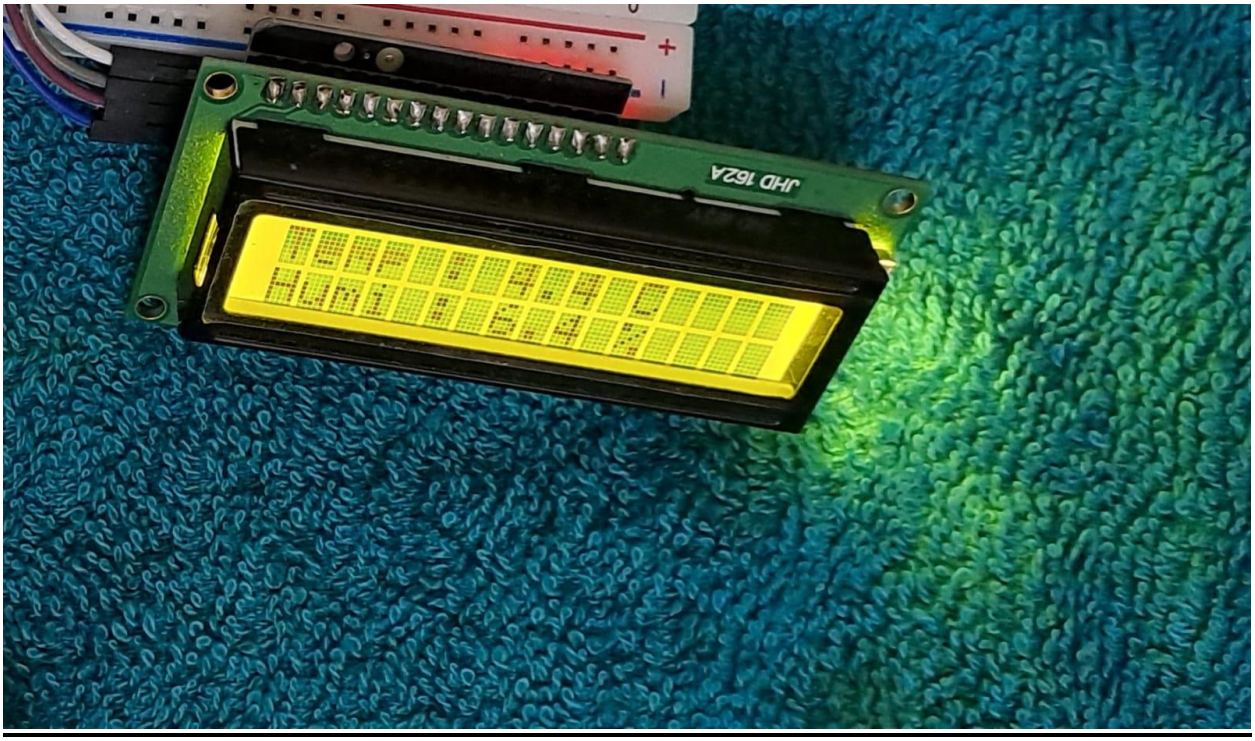
I2C:

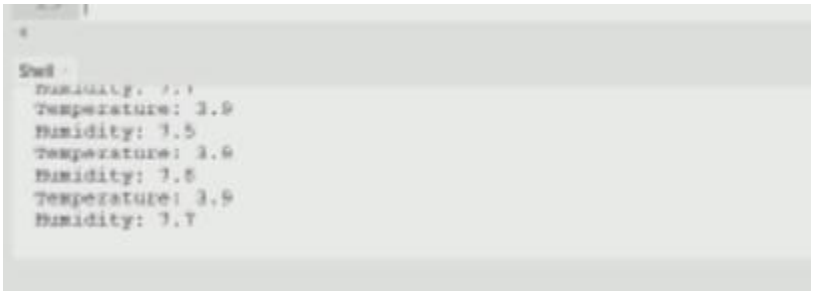

Thonny - C:\Users\Dell\Desktop\Weather Station\pico_i2c_lcd.py @ 77 : 59
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
from lcd_api import LcdApi
from machine import I2C
from time import sleep_ms
DEFAULT_I2C_ADDR = 0x27
# Defines shifts or masks for the various LCD line attached to the PCF8574
MASK_RS = 0x01
MASK_RW = 0x02
MASK_E = 0x04
SHIFT_BACKLIGHT = 3
SHIFT_DATA = 4
class I2cLcd(LcdApi):
"""Implements a character based lcd connected via PCF8574 on i2c."""
def __init__(self, i2c, i2c_addr, num_lines, num_columns):
self.i2c = i2c
self.i2c_addr = i2c_addr
self.i2c.writeto(self.i2c_addr, bytearray([0]))
sleep_ms(20) # Allow LCD time to powerup
# Send reset 3 times
self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
sleep_ms(5) # need to delay at least 4.1 msec
self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
sleep_ms(1)
self.hal_write_init_nibble(self.LCD_FUNCTION_RESET)
sleep_ms(1)
# Put LCD into 4 bit mode
self.hal_write_init_nibble(self.LCD_FUNCTION)
sleep_ms(1)
MicroPython (Raspberry Pi Pico) • COM5
70°F
Mostly cloudy
Search
ENG
IN
23:36
07-12-2022


Thonny - C:\Users\Dell\Desktop\Weather Station\pico_i2c_lcd.py @ 77 : 59
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
sleep_ms(1)
LcdApi.__init__(self, num_lines, num_columns)
cmd = self.LCD_FUNCTION
if num_lines > 1:
cmd |= self.LCD_FUNCTION_2LINES
self.hal_write_command(cmd)
def hal_write_init_nibble(self, nibble):
"""Writes an initialization nibble to the LCD.
This particular function is only used during intiialization.
"""
byte = ((nibble >> 4) & 0x0f) << SHIFT_DATA
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
def hal_backlight_on(self):
"""Allows the hal layer to turn the backlight on."""
self.i2c.writeto(self.i2c_addr, bytearray([1 << SHIFT_BACKLIGHT]))
def hal_backlight_off(self):
"""Allows the hal layer to turn the backlight off."""
self.i2c.writeto(self.i2c_addr, bytearray([0]))
def hal_write_command(self, cmd):
"""Writes a command to the LCD.
Data is latched on the falling edge of E.
"""
byte = ((self.backlight << SHIFT_BACKLIGHT) | (((cmd >> 4) & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
byte = ((self.backlight << SHIFT_BACKLIGHT) | ((cmd & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
MicroPython (Raspberry Pi Pico) • COM5
70°F Mostly cloudy
Search
ENG IN
23:36
07-12-2022


Thonny - C:\Users\Dell\Desktop\Weather Station\pico_i2c_lcd.py @ 77 : 59
File Edit View Run Tools Help
<untitled> main.py pico_i2c_lcd.py dht.py lcd_api.py
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
def hal_backlight_on(self):
"""Allows the hal layer to turn the backlight on."""
self.i2c.writeto(self.i2c_addr, bytearray([1 << SHIFT_BACKLIGHT]))
def hal_backlight_off(self):
"""Allows the hal layer to turn the backlight off."""
self.i2c.writeto(self.i2c_addr, bytearray([0]))
def hal_write_command(self, cmd):
"""Writes a command to the LCD.
Data is latched on the falling edge of E.
"""
byte = ((self.backlight << SHIFT_BACKLIGHT) | (((cmd >> 4) & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
byte = ((self.backlight << SHIFT_BACKLIGHT) | ((cmd & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
if cmd <= 3:
# The home and clear commands require a worst case delay of 4.1 msec
sleep_ms(5)
def hal_write_data(self, data):
"""Write data to the LCD."""
byte = (MASK_RS | (self.backlight << SHIFT_BACKLIGHT) | (((data >> 4) & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
byte = (MASK_RS | (self.backlight << SHIFT_BACKLIGHT) | ((data & 0x0f) << SHIFT_DATA))
self.i2c.writeto(self.i2c_addr, bytearray([byte | MASK_E]))
self.i2c.writeto(self.i2c_addr, bytearray([byte]))
MicroPython (Raspberry Pi Pico) • COM5
70°F Mostly cloudy
Search
ENG IN
23:37
07-12-2022

OUTCOMES:





APPLICATIONS:

There are more applications where the weather monitoring system is very essential and profitable.

- Solar power plant and solar project
- Wind plant
- Conservation engineering
- Aviation and meteorological operations
- Environmental education
- Weather information service

- Disaster mitigation
- Fire station
- Water management and wastewater treatment

The automatic and intelligent weather monitoring system is very versatile. This system allows users to examine atmospheric data that is essential for their operations or projects.

There are more advantages and uses over wireless weather monitoring systems and stations. This system is used also for precision farming (or agricultural) technology in India.

CONCLUSION :

The application is useful to keep track of weather, and the weather station can connect to the internet by adding it to a scientific community made up of several weather stations built from Raspberry Pi+ Sense Hat to monitor weather around the world. From the experiments we noticed small differences between the device and the thermometers used as a reference point, such as: at a temperature of 4.5% error and quite high humidity differences with a 30% error in some experiments, but it is in generally 3-5% error rate.