Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Fundamentals of Digital and Microprocessor Systems

Project report

Students:
Dmytro Kalashnyk
Volodymyr Nashkerskyi
Group: E RBu-2

Kaunas 2024

Introduction

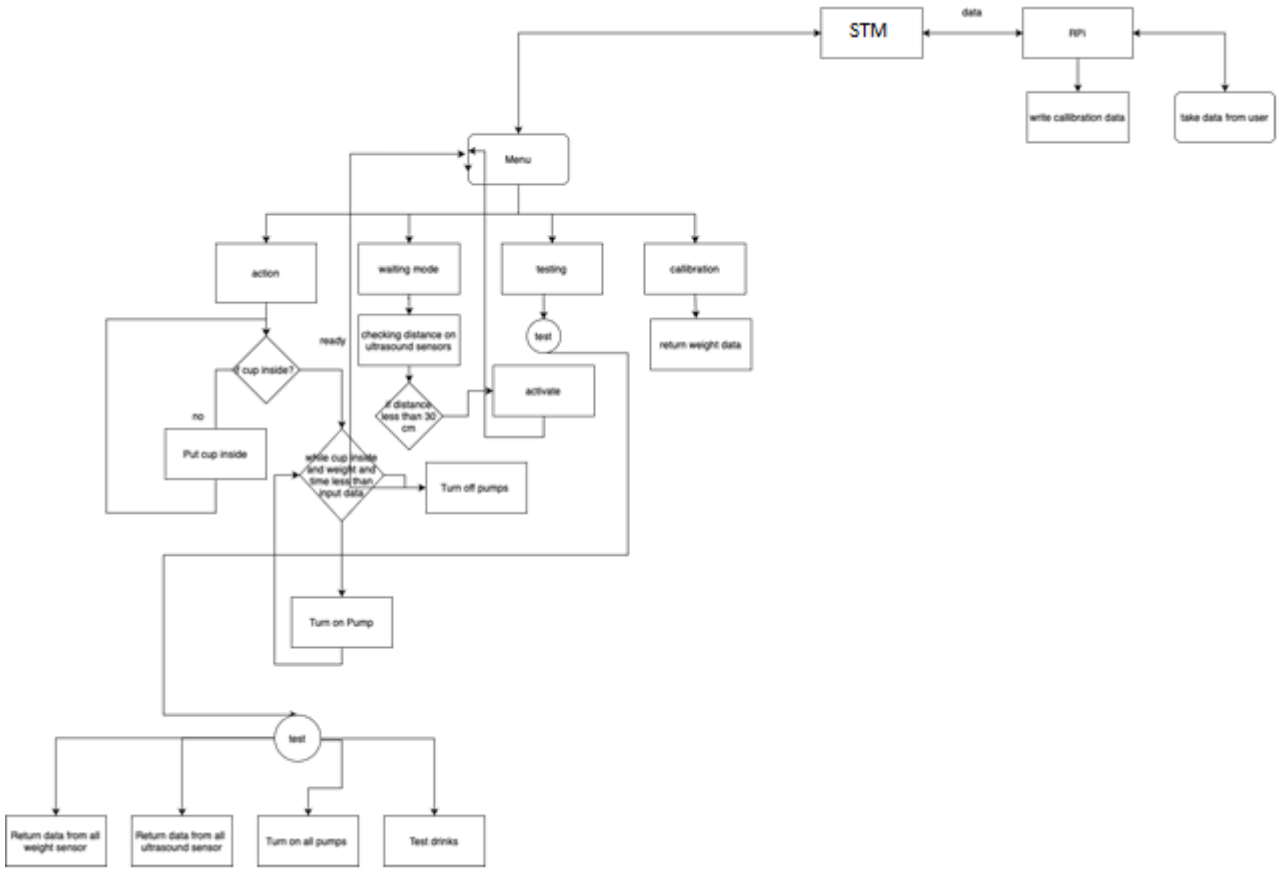
Automated control system (abbreviated as ACS) is a complex of hardware and software, as well as personnel, designed to control various processes within a technological process, production, enterprise. ACS are used in various industries, energy, transport, etc. The term "automated", in contrast to the term "automatic", emphasizes the preservation of some functions by the human operator, either of the most general, goal-setting nature, or not amenable to automation.

Automatic control has been fundamental to the development of automation. Its origins lie in the level control, water clocks, and pneumatics / hydraulics of the ancient world. From the 17th century onwards, systems were designed for temperature control, the mechanical control of mills, and the regulation of steam engines. During the 19th century it became increasingly clear that feedback systems were prone to instability. A stability criterion was derived independently towards the end of the century by Routh in England and Hurwitz in Switzerland. The 19th century, too, saw the development of servomechanisms, first for ship steering and later for stabilization and autopilots. The invention of aircraft added (literally) a new dimension to the problem. Based on servo and communications engineering developments of the 1930s, and driven by the need for high-performance gun control systems, the coherent body of theory known as ‘classical control’ emerged during and just after WW2 in the US, UK and elsewhere, as did cybernetics ideas. Meanwhile, an alternative approach to dynamic modelling had been developed in the USSR based on the approaches of Poincaré and Lyapunov. Information was gradually disseminated, and ‘state-space’ or ‘modern control’ techniques, fuelled by Cold War demands for missile control systems, rapidly developed in both East and West. The immediate post-war period was marked by great claims for automation, but also great fears, while the digital computer opened new possibilities for automatic control.

Project Goal

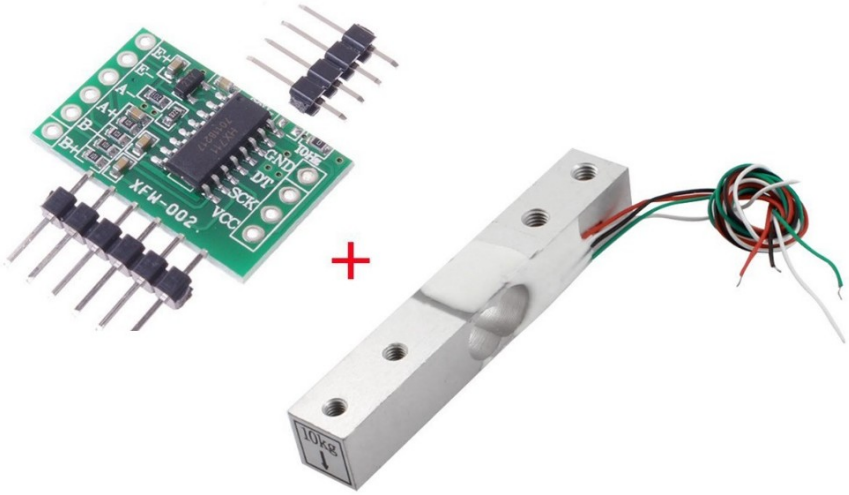
The primary objective of this project is to gain a comprehensive understanding of how to effectively connect an STM microcontroller, Raspberry Pi, weight sensor, and ultrasonic sensor. Through this integration, we aim to demonstrate the proper techniques for measuring and calibrating the values obtained from these sensors. This includes detailing the connection processes, programming necessary interfaces, and implementing calibration procedures to ensure accurate and reliable data collection.

Schema of code:

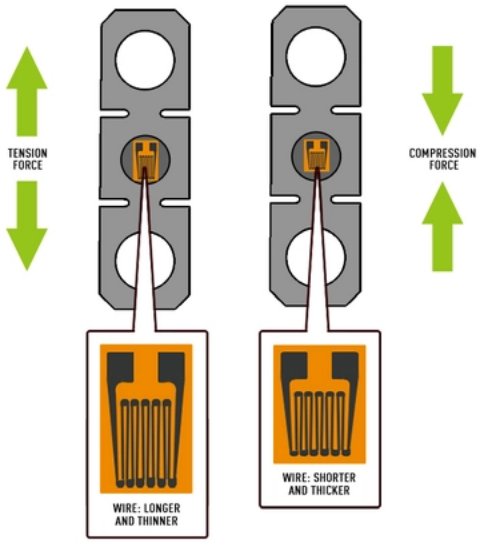


Equipment

5KG Scale Load Cell Weight Weighing Sensor HX711 Weighing Sensors

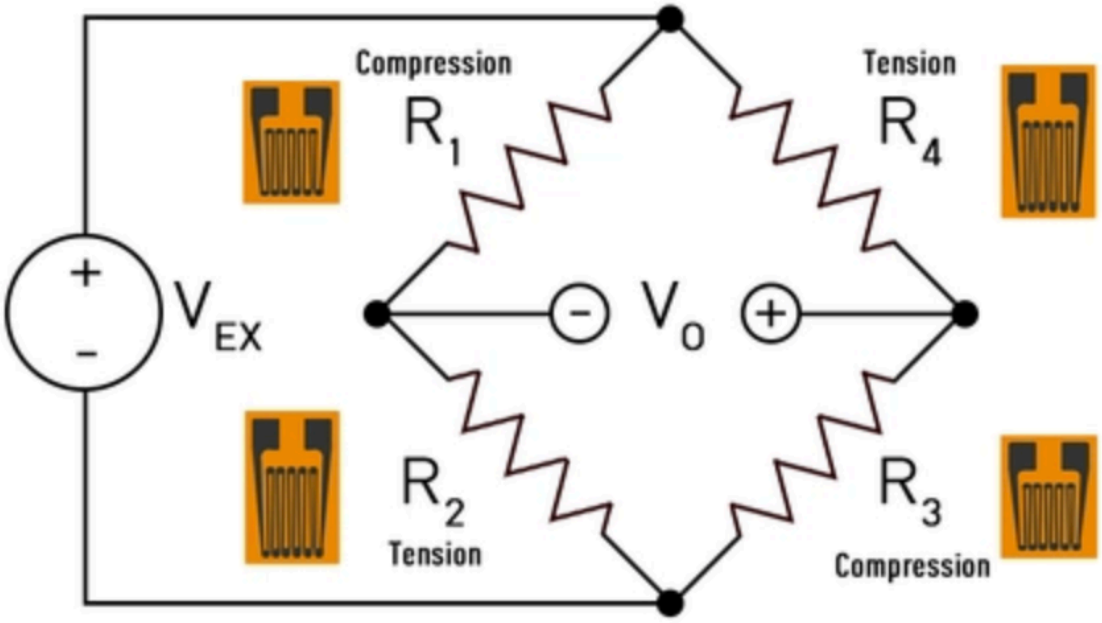


Load cell is measuring changes of the tension in the material by the change of voltage, which is changing because of changing of the resistance inside the material while its go longer and thinner. The magnitude of the electrical signal is directly correlated to the magnitude of the force exerted on the load cell.



Load cell consist of a rigid metal body and when a force is exerted on the load cell, the spring element undergoes slight deformation, which it promptly rebounds from due to its elastic properties.As the shape of the spring element alters, the shape of the strain gauges affixed to it changes as well. Consequently, the electrical resistance of the strain gauges either increases or decreases. By passing an electric current through the strain gauges, the variation in resistance is reflected in the measured voltage output. Since this change in output is directly proportional to the applied weight, the weight of the object can be determined based on the observed voltage change.

$$V_O = \left[\frac{R_3}{(R_3 + R_4)} - \frac{R_2}{(R_1 + R_2)} \right] \times V_{EX}$$



Description of weighting sensors

This module uses 24 high-precision A / D converter chip hx711, is designed for high-precision electronic scale and design, with two analog channel input, the internal integration of a gain programmable amplifier 128. Input circuit can be configured to provide an electrical bridge bridge voltage (such as pressure,weight) sensor model is an ideal high-precision, low-cost sampling front-end module

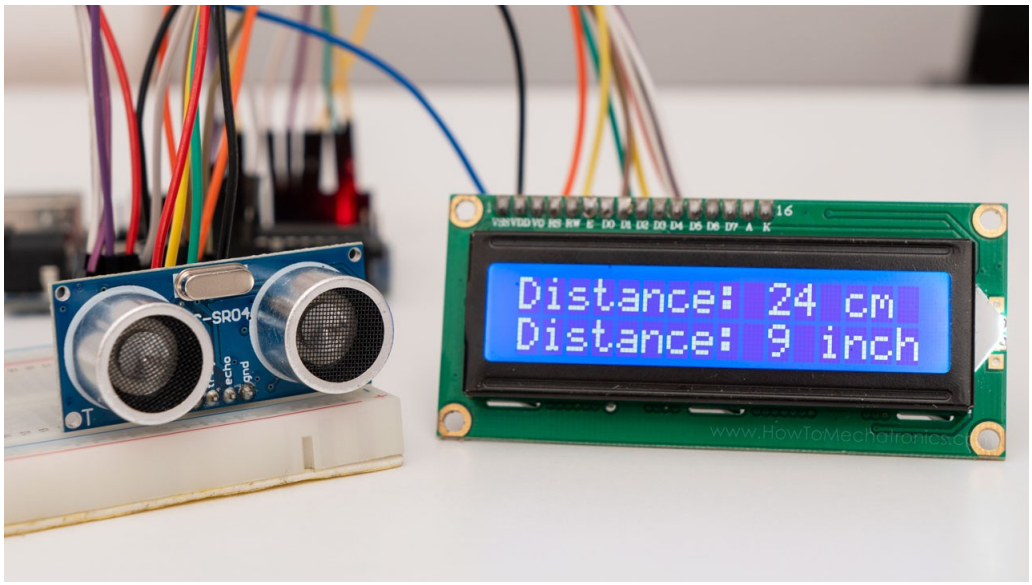
Load cell is measuring changes of the tension in the material by the change of voltage, which is changing because of changing of the resistance inside the material while its go longer and thinner. The magnitude of the electrical signal is directly correlated to the magnitude of the force exerted on the load cell.

HX711/load cell characteristics

- Two selectable differential input channels
- On-chip active low noise PGA with selectable gain of 32, 64 and 128
- On-chip power supply regulator for load-cell and ADC analog power supply
- On-chip oscillator requiring no external component with optional external crystal
- On-chip power-on-reset
- Simple digital control and serial interface: pin-driven controls, no programming needed

- Selectable 10SPS or 80SPS output data rate
- Simultaneous 50 and 60Hz supply rejection
- Current consumption including on-chip analog power supply regulator: normal operation < 1.5mA, power down < 1uA
- Operation supply voltage range: 2.6 ~ 5.5V
- Operation temperature range: -40 ~ +85℃
- 16 pin SOP-16 package

Ultrasonic sensors

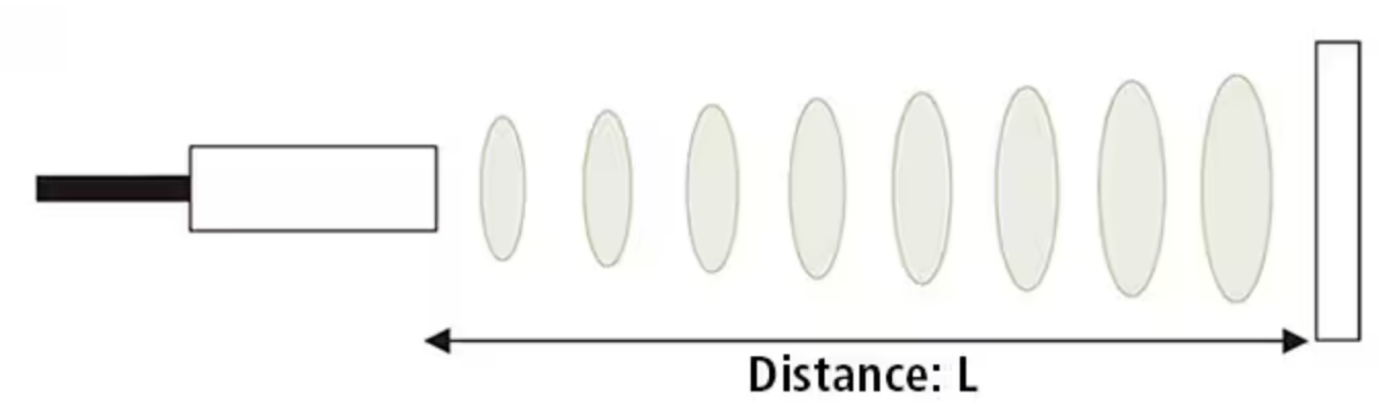


Ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. We measure the distance to the target by measuring the time between the emission and reception and then apply the formula from datasheet or from sonic speed calculation.

Ultrasonic sensing is one of the best ways to sense proximity and detect levels with high reliability. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing.

The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

The working principle of this module is simple. It sends an ultrasonic pulse out at 40kHz which travels through the air and if there is an obstacle or object, it will bounce back to the sensor. By calculatingthe travel time and the speed of sound, the distance can be calculated.

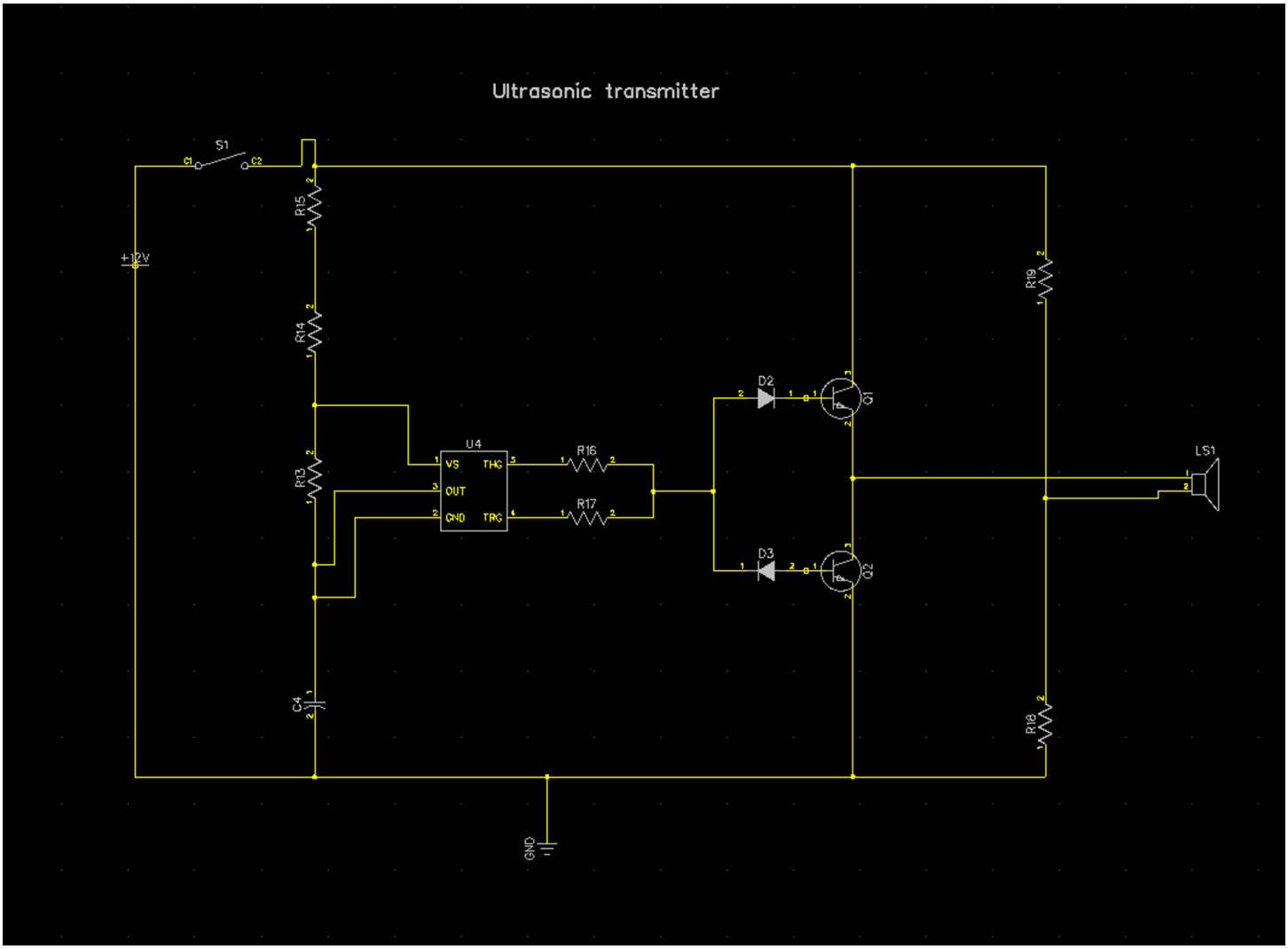


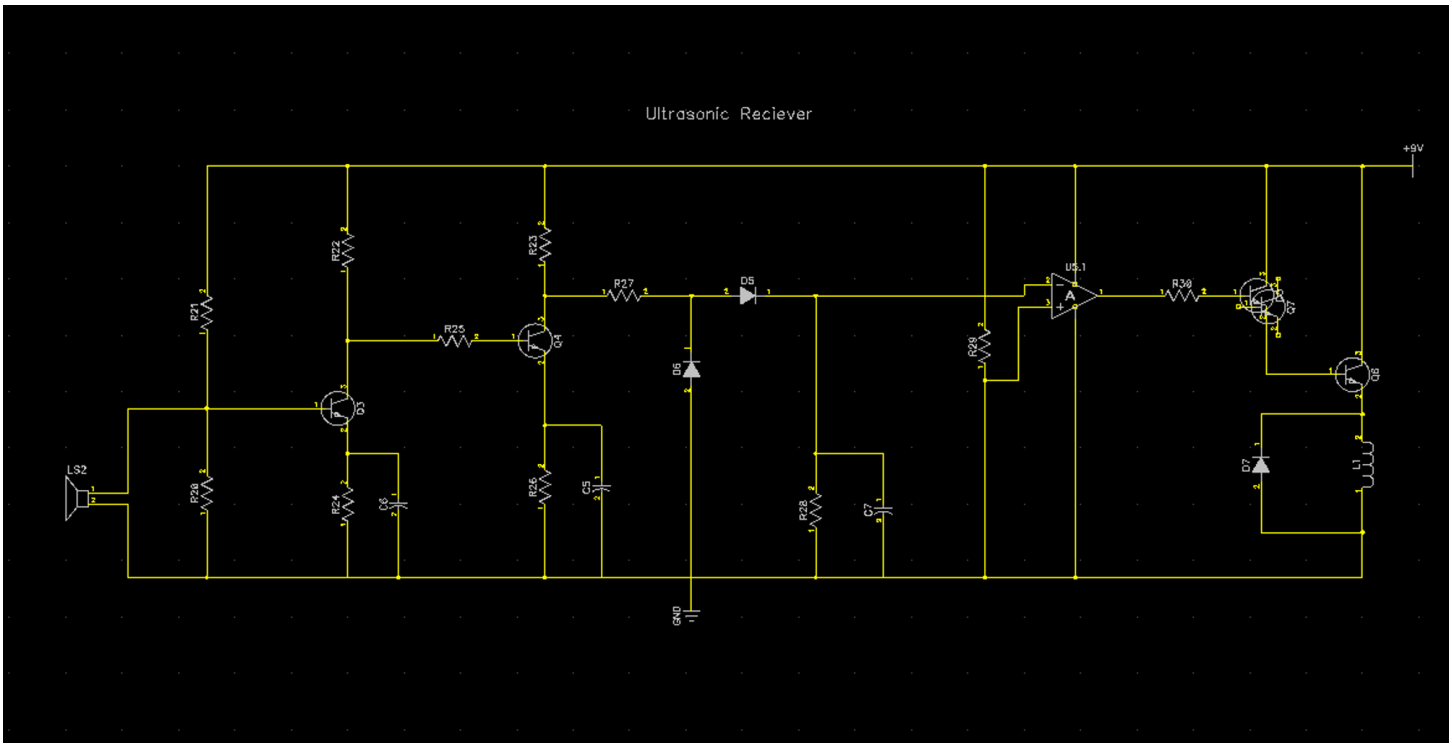
Distance L = 1/2 × T × C

T - time, C - sonic speed. We are dividing it by 2 cause we are measuring time for go and return distance.

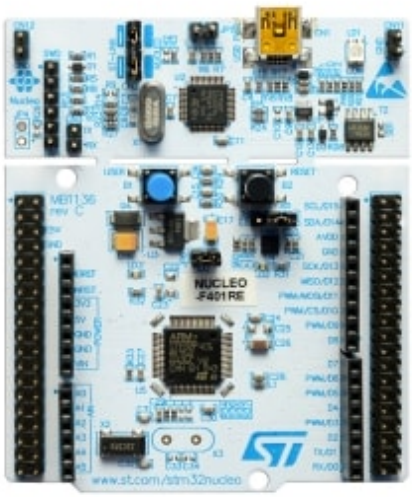
So, firsly we turn on trigger pin which create ultrasonic wave. Then we are waiting for 10 ms, cause it needs time for creation. Than we turn off trigger pin and reading echo pin which which will give us a signal when the sound went back. While waiting for back signal measuring time.

Ultrasonic schematics:


Ultrasonic transmitter
S1
C1
C2
+12V
R15
R14
R13
C4
U4
VS
OUT
GND
THG
TRG
R16
R17
D2
D3
Q1
Q2
R19
R18
LS1
GND


Ultrasonic Reciever
+9V
LS2
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
C5
C6
C7
D5
D6
D7
Q5
Q6
Q7
Q8
U5:1
L1
GND

STM Nucleo-64 F401RE



The STM32 Nucleo-64 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.
The ARDUINO® Uno V3 connectivity support and the ST morpho headers allow the easy expansion of the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.

The STM32 Nucleo-64 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.

The STM32 Nucleo-64 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

All features
Common features
STM32 microcontroller in an LQFP64 or LQFP48 package
1 user LED shared with ARDUINO®
1 user and 1 reset push-buttons
32.768 kHz crystal oscillator
Board connectors:
ARDUINO® Uno V3 expansion connector
ST morpho extension pin headers for full access to all STM32 I/Os
Flexible power-supply options: ST-LINK USB VBUS or external sources
Comprehensive free software libraries and examples available with the STM32Cube MCU Package
Support of a wide choice of Integrated Development Environments (IDEs) including IAR Embedded Workbench®, MDK-ARM, and STM32CubeIDE

Features specific to some of the boards

External or internal SMPS to generate Vcore logic supply:NUCLEO-L412RB-P, NUCLEO-L433RC-P, NUCLEO-L452RE-P, and NUCLEO-U545RE-Q

24 MHz or 48 MHz HSE:NUCLEO-C031C6, NUCLEO-G431RB, NUCLEO-G474RE, and NUCLEO-G491RE

User USB Device full speed, or USB SNK/UFP full speed:NUCLEO-H503RB, NUCLEO-H533RE, and NUCLEO-U545RE-Q

Cryptography:NUCLEO-H533RE, NUCLEO-U083RC, and NUCLEO-U545RE-Q

Board connectors:

External SMPS experimentation dedicated connector:NUCLEO-L412RB-P, NUCLEO-L433RC-P, NUCLEO-L452RE-P, and NUCLEO-U545RE-Q

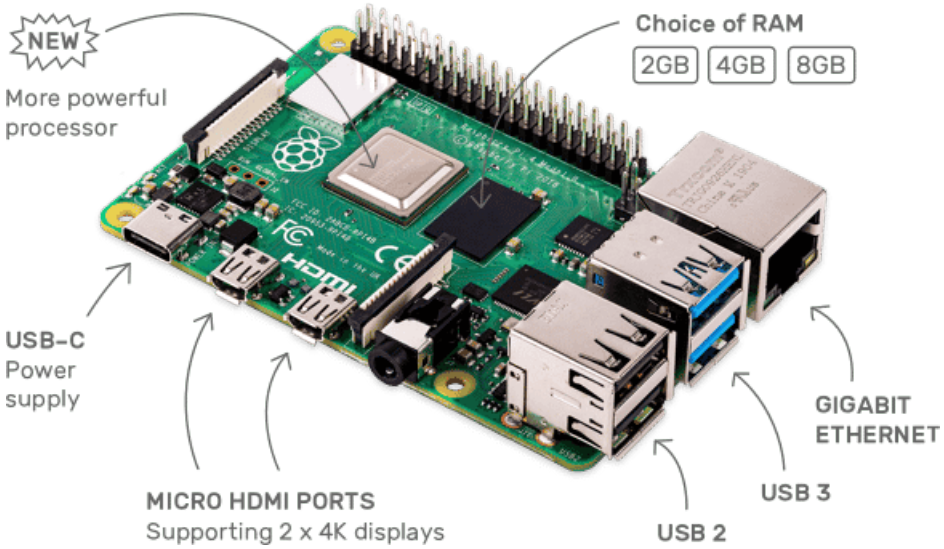
USB Type-C®, Micro-B, or Mini-B connector for the ST-LINK

USB Type-C® user connector:NUCLEO-H503RB, NUCLEO-H533RE, and NUCLEO-U545RE-Q

MIPI® debug connector:NUCLEO-G431RB, NUCLEO-G474RE, NUCLEO-G491RE, NUCLEO-H503RB, NUCLEO-H533RE, NUCLEO-U031R8, and NUCLEO-U083RC

On-board ST-LINK (STLINK/V2-1, STLINK-V3E, STLINK-V2EC, or STLINK-V3EC) debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port.

Raspberry Pi 4 Model B



Raspberry Pi is a series of microcomputer models that can connect to standard monitors and peripherals for various applications. RP models are slightly bigger than a credit card and contain hardware components to support traditional desktop uses like file creation, storage, and internet streaming.
Raspberry Pi models exemplify a system-on-chip (SoC) where the microdevice contains a single integrated circuit. RP models include a central processing unit (CPU), power supply, USB ports, and RAM to support common computer use cases.

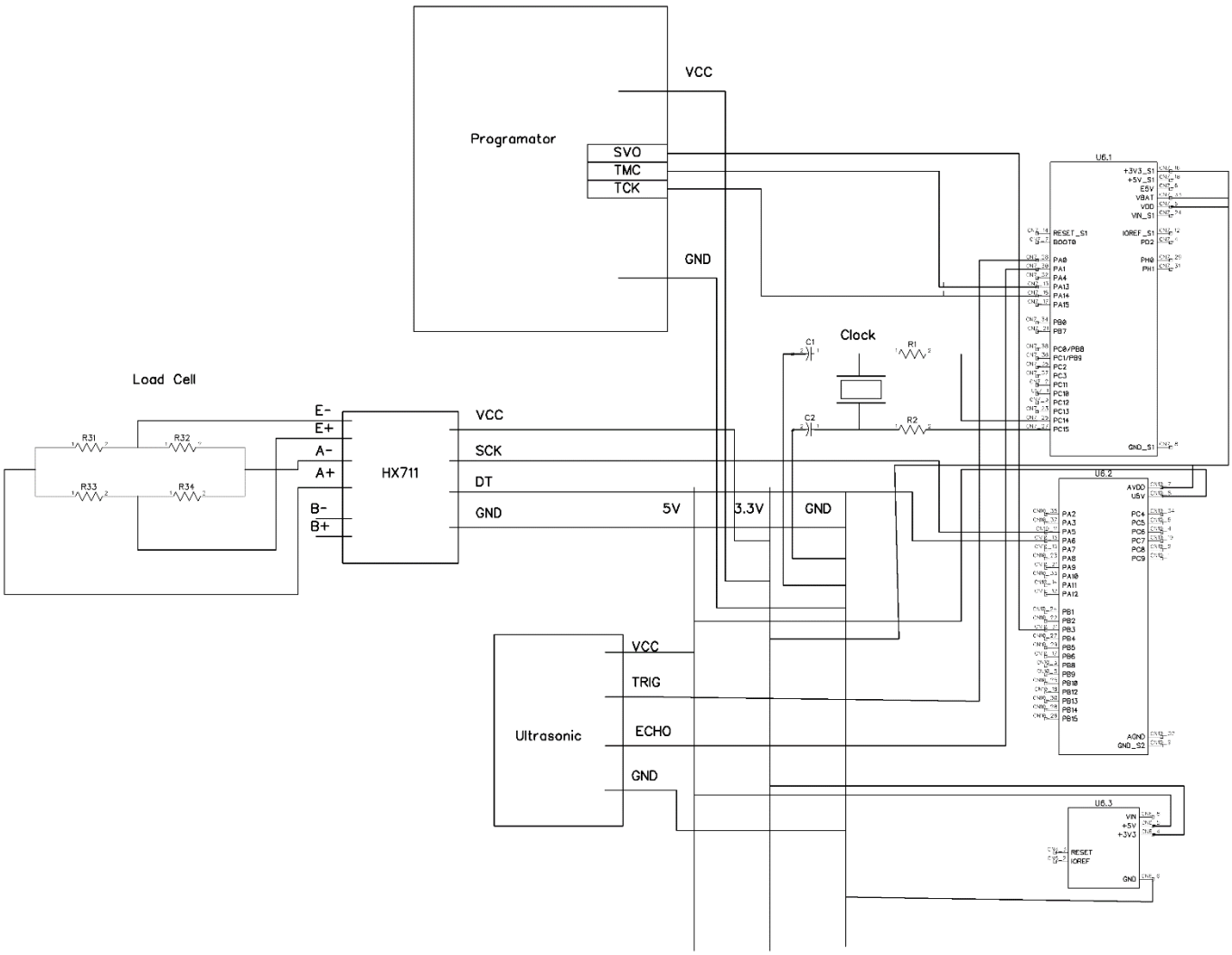
Customers can use their RP model through an existing monitor, keyboard, or mouse with an SD card for storage and appropriate connecting cables.

Specifications

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.1, Vulkan 1.0
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

* A good quality 2.5A power supply can be used if downstream USB peripherals consume less than 500mA in total.

System schematics design



Relevance of Engineering Solutions

Industry Use Cases

Manufacturing In the manufacturing industry, integrated systems comprising microcontrollers, single-board computers, weight sensors, and ultrasonic sensors are crucial for automated weighing and distance measurement. These systems ensure that materials and products are accurately weighed and positioned during production. For instance, in an assembly line, weight sensors can monitor the weight of components to ensure consistency and quality, while ultrasonic sensors measure the distance and alignment of parts, preventing misplacements and enhancing the precision of robotic arms.

Logistics In the logistics sector, such systems are vital for optimizing the loading and spacing of goods. Weight sensors are used to verify the weight of packages, ensuring they comply with shipping regulations and preventing overloading. Ultrasonic sensors assist in measuring the distance between packages, enabling optimal packing and maximizing the use of storage space in warehouses and transport vehicles. This ensures safe and efficient transportation of goods, reduces the risk of damage, and improves overall logistics efficiency.

Benefits

Efficiency These systems improve efficiency by speeding up processes and reducing downtime. In manufacturing, they ensure consistent and fast production rates. In logistics, they optimize the loading and unloading processes, saving time and resources. In healthcare, they enable quick and accurate patient monitoring, freeing up medical staff for other critical tasks.

Real-Time Data Processing The integration of microcontrollers and single-board computers enables real-time data processing, which is crucial for timely decision-making. In manufacturing, real-time data allows for immediate adjustments in production. In logistics, it helps in tracking and managing inventory efficiently. In healthcare, real-time monitoring ensures that any changes in a patient's condition are promptly detected and addressed, potentially saving lives.

Alternative Hardware Solutions

Microcontroller Options

STM Microcontrollers STM microcontrollers, particularly the STM32 series, are known for their high performance, low power consumption, and a wide range of peripherals. They offer various models to cater to different applications, from basic tasks to complex real-time processing. STM microcontrollers are popular in industrial and professional applications due to their robustness and extensive feature set.

Arduino Arduino microcontrollers are widely used for educational purposes, prototypes, and hobbyist projects. They are known for their simplicity and ease of use, with an extensive community and a vast range of libraries and tutorials available. While Arduino boards are generally less powerful than STM microcontrollers, they are sufficient for many basic to intermediate projects.

ESP32 The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth capabilities. It offers excellent performance and a wide range of features at a relatively low cost. The ESP32 is ideal for IoT

applications, thanks to its wireless connectivity and extensive support from the community. It also has a dual-core processor, which provides additional processing power compared to typical Arduino boards.

Single-board Computers

Raspberry Pi The Raspberry Pi is a versatile and popular single-board computer known for its low cost, robust community support, and wide range of compatible peripherals. It is widely used in educational settings, DIY projects, and even some industrial applications. Raspberry Pi supports various operating systems, including Raspbian (Raspberry Pi OS), Ubuntu, and even Windows IoT Core.

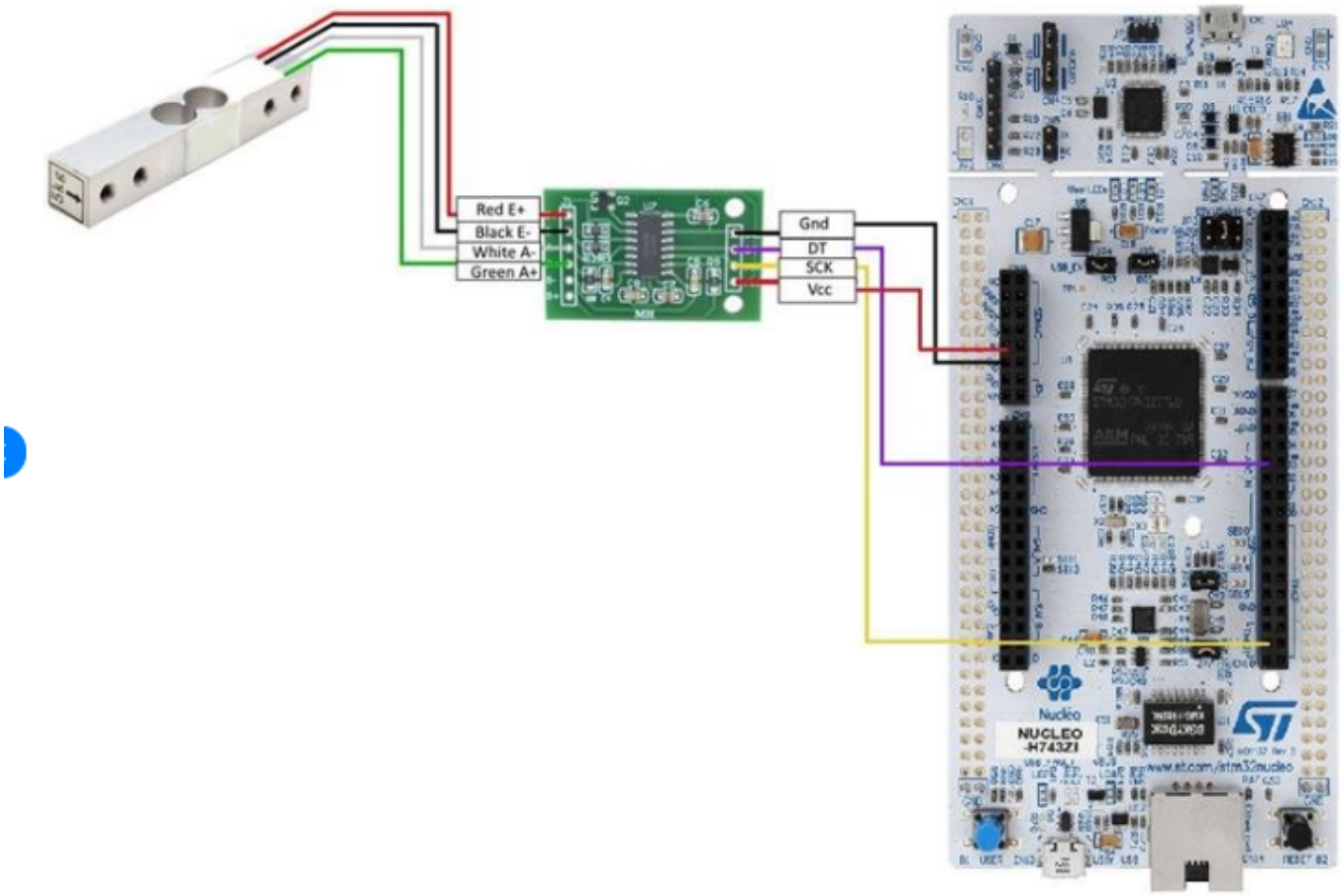
BeagleBone BeagleBone boards, such as the BeagleBone Black, are powerful single-board computers that offer excellent I/O capabilities and are favored for more complex and industrial applications. They provide better real-time processing capabilities compared to Raspberry Pi and have strong community support, although not as extensive as the Raspberry Pi.

Jetson Nano The NVIDIA Jetson Nano is designed for AI and machine learning applications, providing powerful GPU capabilities for edge computing. It excels in tasks requiring heavy computation, such as computer vision and deep learning, and supports a range of AI frameworks. The Jetson Nano is more expensive than the Raspberry Pi but offers superior processing power for specific use cases.

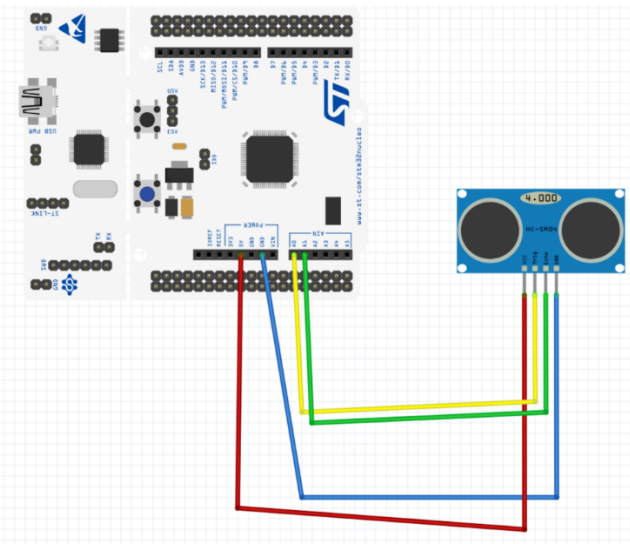
Odroid Odroid boards, like the Odroid-XU4, offer high performance with multi-core processors and are suitable for applications requiring significant computing power. They support a variety of operating systems and have a good balance of performance and cost. Odroid boards have a smaller community compared to Raspberry Pi but are still well-supported.

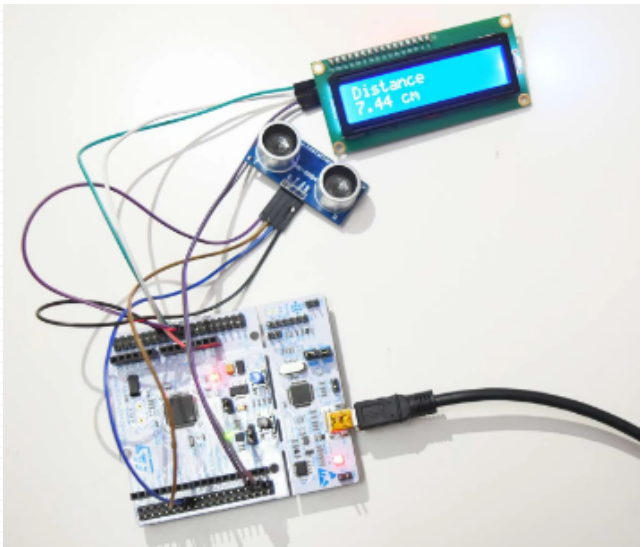
Connectivity of elements

Weight sensors:



Ultrasonic sensors





Software code:

https://github.com/volodicd/Microproc_Project

hx711.h

```
typedef struct{
        GPIO_TypeDef* dout_port;
        uint16_t dout_pin;
        GPIO_TypeDef* sck_port;
        uint16_t sck_pin;
        int32_t zeroOffset;
        double scaleFactor;
}
```

Structure definition

HX711_Init

This function initializes the HX711 structure by setting its data (DOUT) and clock (SCK) pin ports and pins. It also resets the zero offset and scale factor to default values, and sets the SCK pin to a low state.

HX711_Read

This function reads a 24-bit value from the HX711 by toggling the SCK pin 24 times and assembling the bits read from the DOUT pin. The final value is adjusted to account for the sign bit and then returned as a 32-bit signed integer.

HX711_Tare

This function tares (calibrates) the HX711 by reading a specified number of values and averaging them to set the zero offset. This helps to eliminate the current weight on the scale to use it as the reference zero.

HX711_GetWeight

This function calculates the weight by averaging a specified number of readings from the HX711, subtracting the zero offset, and then dividing by the scale factor. The result is returned as a double representing the measured weight.

funcs.c

```
typedef struct Time{
        uint8_t sec;
        uint8_t min_S;
        uint8_t hour;
}Time;

typedef struct SensorMes{
        double dist;
        int weight;
```

Time meas_time;

}SensorMes;

string_length

This function calculates the length of a given string by iterating through each character until it encounters the null terminator (`\0`). It returns the length as a `size_t` value.

save_measurement_to_flash

It defines a static variable `flash_address` starting at `FLASH_USER_START_ADDR`. It then calls `Flash_Write` to write the `meas_data` to flash memory at the current `flash_address`. After writing, it increments `flash_address` by the size of `meas_data` to prepare for the next write.

read_all_measurements_from_flash

It calculates the size of each `SensorsMes` record. It then iterates through `max_records`, reading each record from flash memory into `data_array` by calling `Flash_Read` for each record's address.

get_flash_end_addr

It calculates the end address by starting at `FLASH_USER_START_ADDR` and adding the total size of all saved measurements (number of measurements `counter` multiplied by the size of `SensorsMes`) minus one.

clean_flash

It calls `Flash_Erase`, passing the start address of the flash memory and the total size of the used flash memory area (calculated using `get_flash_end_addr`).

string_check

This function compares two strings to check if they are identical. It returns 1 if the strings are the same length and have identical characters, and 0 otherwise.

send_data_huart

This function sends sensor data via UART, including the distance and weight measurements along with the current time. It formats and transmits a message, and then saves the measurement data to EEPROM, incrementing a global counter.

tim_spec_del

This function creates a delay for a specified number of microseconds using a hardware timer (TIM2). It resets the timer counter and waits until the counter reaches the specified value.

read_ultrasonic

This function reads the time taken for an ultrasonic pulse to travel to an object and back, measuring the duration the echo pin remains high. The measured time is used to calculate the distance.

meas_dist

This function measures the distance using the ultrasonic sensor by calling `read_ultrasonic` and converting the pulse duration to distance in centimeters. It introduces a delay of 1 second before the measurement.

SensorsMes * find_max

This function finds the maximum distance and weight from an array of sensor measurements. It iterates through the measurements and keeps track of the highest values, returning a dynamically allocated structure containing the maximum values.

SensorsMes * find_min

This function finds the minimum distance and weight from an array of sensor measurements. It iterates through the measurements and keeps track of the lowest values, returning a dynamically allocated structure containing the minimum values.

menu

The `menu` function receives commands from a UART interface and processes them to perform specific actions. If the command is "max," it retrieves the maximum measurement data from flash memory and transmits it via UART. Similarly, for the "min" command, it retrieves the minimum measurement data. If the command is "clean," it erases the flash memory area used for storing measurements.

flash.c

Flash_Write

This function writes an array of bytes to flash memory starting at the specified address. It unlocks the flash memory for writing, writes each byte individually using `HAL_FLASH_Program`, and then locks the flash memory again to prevent further modifications.

Flash_Read

This function reads an array of bytes from flash memory starting at the specified address. It copies each byte from the flash memory to the provided data buffer.

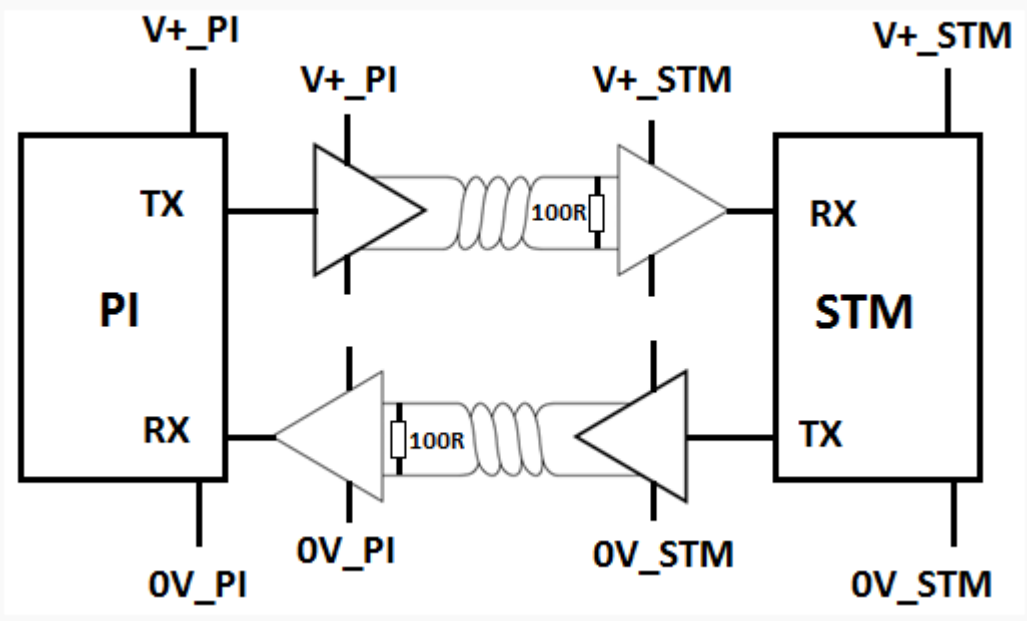
Flash_Erase

This function erases a specified range of flash memory starting at `startAddress` and covering `length` bytes. It calculates the start and end sectors, unlocks the flash memory, erases the sectors using `HAL_FLASHEx_Erase`, and then locks the flash memory.

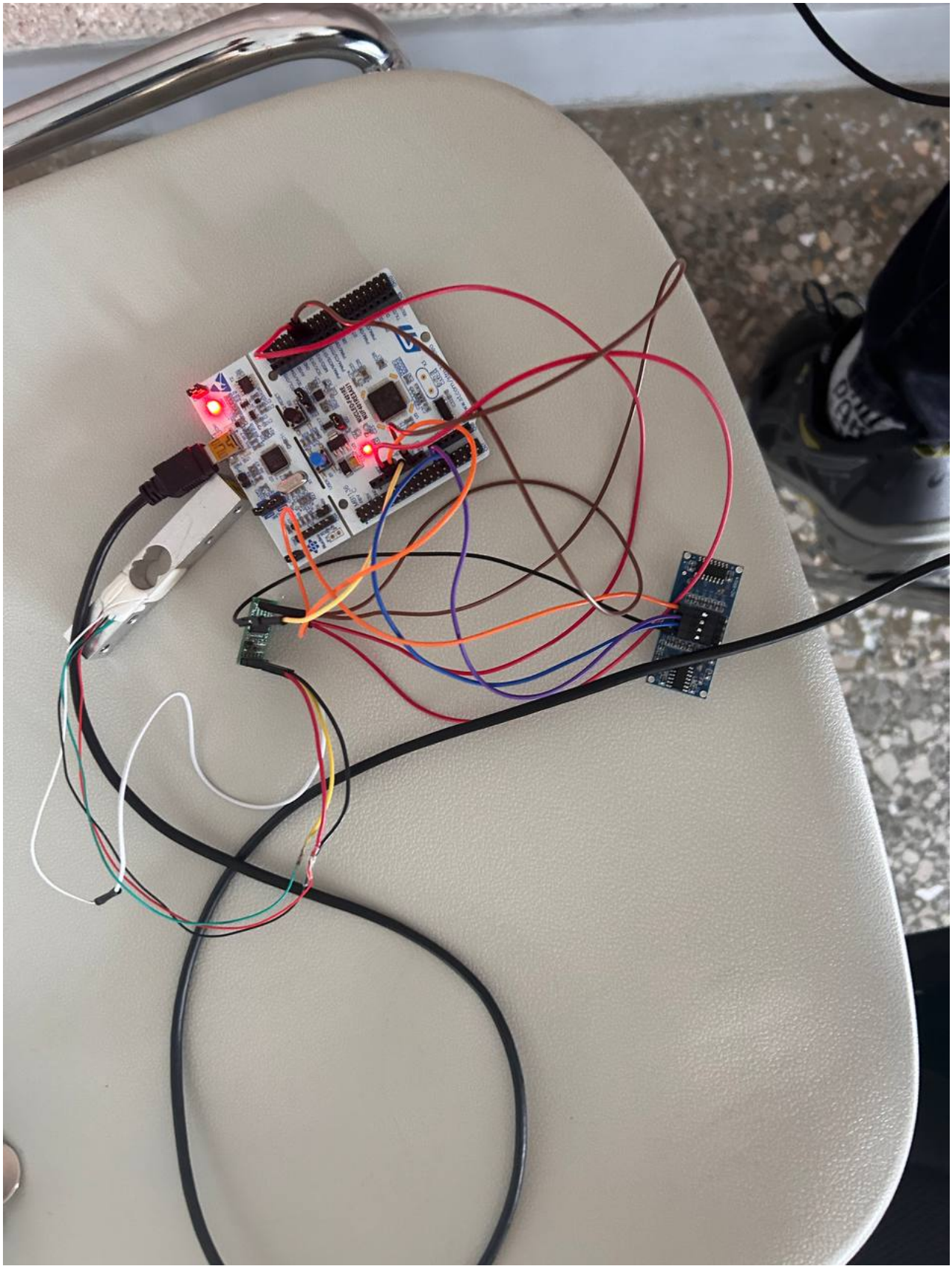
GetSector

This helper function determines which flash sector corresponds to a given memory address. It returns the sector number based on the address range comparisons.

STM32 and RPI



Result



Conclusion

In this project, we successfully integrated an STM microcontroller, a Raspberry Pi, a weight sensor, and an ultrasonic sensor to create a comprehensive measurement and calibration system. Understanding the effective connection techniques, programming interfaces and calibration procedures required for accurate data collection from these sensors has been the primary objective.

Key Achievements:

1. Hardware integration: We've set up safe and efficient connections between the STMC microcontroller, Raspberry Pi, weight sensors or ultrasonic sensors. The appropriate interfacesI2C, SPI, UART or GPIO were used to connect each component and the requirements for power supply have been adequately met.
2. Programming Interfaces: The STM microcontroller was programmed to interface with the sensors, capturing raw data and transmitting it to the Raspberry Pi. The Raspberry Pi handled data storage, processing, and other communication functions as a central processing unit.
3. Sensor calibration: For both the weight and ultrasonic sensors, specific calibration procedures have been introduced. In order to ensure the accuracy and reliability of measurements, this included zeroing, scale or compensation for environmental factors.
4. Data collection and analysis: the system has shown that data can be collected efficiently, with Raspberry Pi being able to process it in order to provide valuable information. The accuracy and consistency of the sensor readings was ensured by the calibration algorithms.

Literature

https://www.amazon.de/-/en/dp/B07MJF9Z4K?psc=1&ref=ppx_yo2ov_dt_b_product_details
https://www.ebay.com/itm/202104396809
https://www.globalspec.com/learnmore/industrial_computers_embedded_computer_components/industrial_computing/relay_boards
https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm#:~:text=An%20ultrasonic%20sensor%20is%20an,information%20about%20an%20object's%20proximity.
https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ https://www.webopedia.com/definitions/raspberry-pi/
https://www.playembedded.org/blog/detecting-obstacles-hc-sr04/
https://forums.raspberrypi.com/viewtopic.php?t=213305
https://community.st.com/t5/stm32-mcus-motor-control/best-communication-protocol-to-connect-stm32-to-a-raspberry-pi/td-p/633590