ABSTRACT

The main objective of this project is to develop a solution for measuring the water level of any tank to help the user get the real-time data which can help them in operating pumps to refill the tank when required as well as to switch it offso that no water gets wasted.

As technology is advancing, problems are also getting smarter. It is not always possible to administer the exact water level and use the pump accordingly. Water tank overflow is therefore prevalent in many houses causing waste of litres of water. This problem is solved by our project.

IOT based smart water management system provide a most modern solution with smartphones as well as web. It let's you know the status of water level and when it's time to refill the water tank. In order to achieve this, an Raspberry Pi Pico W module, an ultrasonic sensor is used. The sensor measures the depth of water tank, and returns the amount of water present in it.

INTRODUCTION

Nowadays, we have remote controls for our television sets and other electronic systems, which have made our lives real easy. In this era of smart gadgets, have you ever wondered about smart water management system which wouldgive the ability to inform the user remotely placed,when the water tank is full, half empty or need a refill.

Off-course, Yes! But, are the available options cost-effective? If the answer is No, we have found a solution to it.

We have come up with a new system called IOT based smart water management system. This system is super-cost effective and can give the user the ability to know the status of a water tank remotely placed,without even spending fora remote control. This project helps the user to know when a refill is needed using his/her smartphone through Firebase cloud service from anywhere.

Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save our time. To save people’s time and the mental anxiety of remembering when to switch on the pump ,we are introducing a Smart Water Management System using IOT. With the help of this system you can keep an eye on the water level and switch on the pump accordingly and switch it off as well without causing any wastage of water, all of this just using your mobile phone.

COMPONENTS USED

1. Raspberry Pi Pico W
2. Breadboard
3. Male to Male wires
4. Male to Female wires
5. Ultrasonic Sensor(HCSR04)
6. Power supply
7. Relay module
8. Water pump
9. Wi-fi enabled device

DESCRIPTION

RASPBERRY PI PICO W

The Raspberry Pi Pico W is a compact and versatile microcontroller board developed by the Raspberry Pi Foundation. It features the RP2040 chip, which includes a dual-core ARM Cortex-M0+ processor running at 133 MHz, and 264KB of SRAM. The "W" in its name signifies built-in Wi-Fi connectivity, which opens up new possibilities for IoT projects and wireless communication. The Pico W also has 2MB of flash memory, a range of GPIO pins for various inputs and outputs, and supports popular programming languages like MicroPython and C/C++. Its small size and low cost make it ideal for embedded projects and prototyping.

FEATURES:

1. RP2040 Chip: Dual-core ARM Cortex-M0+ processor running at 133 MHz.
2. Memory: 264KB SRAM for fast access.
3. Flash Storage: 2MB QSPI flash memory for program and data storage.
4. Wi-Fi: Integrated 2.4 GHz wireless networking with the Infineon CYW43439 chip.
5. GPIO Pins: 26 multi-function GPIO pins with 3.3V logic levels.
6. PWM: Supports Pulse Width Modulation for various control applications.
7. ADC: Three 12-bit Analog-to-Digital Converters for analog input.
8. Serial Interfaces: UART, SPI, and I2C communication protocols.
9. I2S: Digital audio interface support.
10. Power Supply: Powered via USB Type-C or external 5V source.
11. Programming: Supports MicroPython, C/C++, and other programming environments.
12. Physical Size: Compact 51 x 21 mm form factor.
13. Mounting: 3.5 mm pitch, 2.54 mm spaced through-hole pin headers.
14. Debug Interface: SWD (Serial Wire Debug) for development and debugging.
15. Onboard LED: For status indication and simple output tasks.

HARDWARE COMPONENTS:

1. RP2040 Microcontroller: Dual-core ARM Cortex-M0+ processor, 133 MHz.
2. Flash Memory: 2MB QSPI flash for program storage.
3. SRAM: 264KB for fast, temporary data storage.
4. Wi-Fi Module: Infineon CYW43439 for 2.4 GHz wireless connectivity.
5. GPIO Pins: 26 multi-function pins with 3.3V logic levels.
6. Analog-to-Digital Converters: Three 12-bit ADC channels.
7. Pulse Width Modulation (PWM): For controlling servos, motors, etc.
8. Serial Interfaces: Multiple UART, SPI, and I2C ports for communication.
9. I2S Interface: For digital audio applications.
10. Power Supply: USB Type-C connector or 5V external input.
11. Onboard LED: For status indication and simple output.
12. Debug Interface: SWD (Serial Wire Debug) for development.
13. Crystal Oscillator: Provides precise clock timing for the microcontroller.
14. Capacitors and Resistors: For signal conditioning and stability.

POWER REQUIREMENTS:

The Raspberry Pi Pico W has the following power requirements:

1. Operating Voltage: 1.8V to 3.3V for the core logic.
2. Power Supply:
   - USB Power: 5V via USB Type-C connector.
   - External Power: 5V through the VSYS pin.
3. Current Consumption:
   - Typical Consumption: Approximately 100-200 mA during normal operation.
   - Low Power Modes: Can be reduced significantly in low-power or sleep modes.
4. Maximum Current Draw: Up to 500 mA through USB, depending on connected peripherals and usage.
5. Power Regulation: Onboard voltage regulators provide stable 3.3V for the RP2040 chip and peripherals.

These requirements ensure that the Pico W can be powered via common USB sources or external 5V supplies while providing flexibility for various applications.

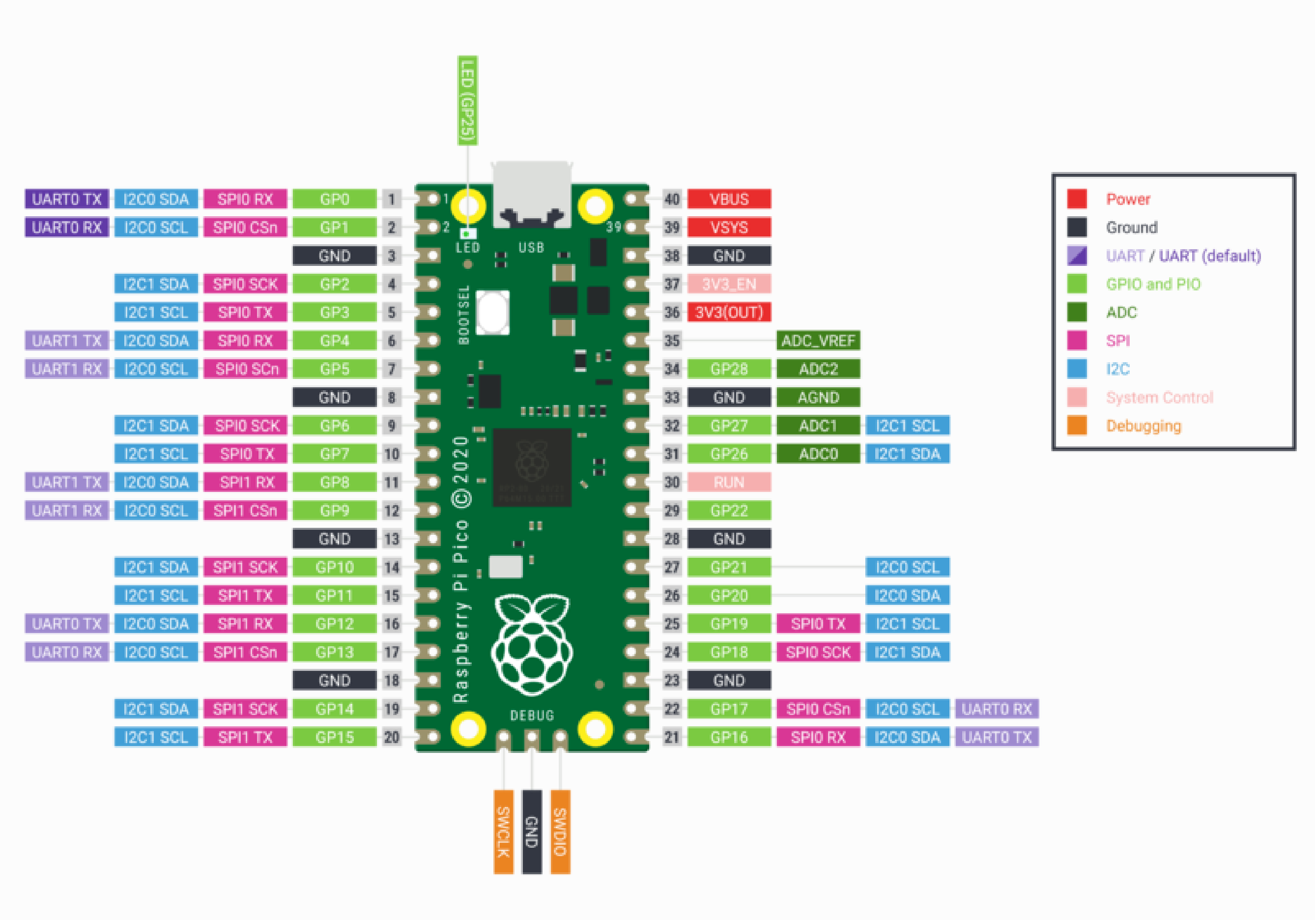
PERIPHERALS AND I/0:

The Raspberry Pi Pico W offers a variety of peripherals and I/O options for interfacing with external components:

1. GPIO Pins: 26 multi-function GPIO pins, which can be used for digital input and output, including:
    - Digital I/O: Configurable as input or output.
    - PWM: Pulse Width Modulation output.
2. Analog-to-Digital Converters (ADC): 3 x 12-bit ADC channels for measuring analog signals.
3. Serial Communication Interfaces:
    - UART: Two UARTs for serial communication.
    - SPI: Two SPI interfaces for high-speed data transfer.
    - I2C: Two I2C interfaces for connecting to I2C devices.
4. I2S Interface: For digital audio data transmission.
5. Power Pins:
    - 3.3V: Regulated power output for powering external components.
    - 5V: Directly from USB or external power source.
    - GND: Ground connections for circuit completion.
6. Onboard LED: Programmable LED for status indications and visual output.
7. SWD Interface: Serial Wire Debug for programming and debugging.
8. External Interrupts: GPIO pins can be configured to trigger interrupts on events.

These peripherals and I/O options make the Pico W highly versatile for a wide range of electronics projects, from simple sensors to complex communication systems.

RASPBERRY PI PICO W ARCHITECTURE:


LED (GP25)
UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1
UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2
GND | 3
I2C1 SDA | SPI0 SCK | GP2 | 4
I2C1 SCL | SPI0 TX | GP3 | 5
UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6
UART1 RX | I2C0 SCL | SPI0 SCn | GP5 | 7
GND | 8
I2C1 SDA | SPI0 SCK | GP6 | 9
I2C1 SCL | SPI0 TX | GP7 | 10
UART1 TX | I2C0 SDA | SPI1 RX | GP8 | 11
UART1 RX | I2C0 SCL | SPI1 CSn | GP9 | 12
GND | 13
I2C1 SDA | SPI1 SCK | GP10 | 14
I2C1 SCL | SPI1 TX | GP11 | 15
UART0 TX | I2C0 SDA | SPI1 RX | GP12 | 16
UART0 RX | I2C0 SCL | SPI1 CSn | GP13 | 17
GND | 18
I2C1 SDA | SPI1 SCK | GP14 | 19
I2C1 SCL | SPI1 TX | GP15 | 20
40 | VBUS
39 | VSYS
38 | GND
37 | 3V3_EN
36 | 3V3(OUT)
35 | ADC_VREF
34 | GP28 | ADC2
33 | GND | AGND
32 | GP27 | ADC1 | I2C1 SCL
31 | GP26 | ADC0 | I2C1 SDA
30 | RUN
29 | GP22
28 | GND
27 | GP21 | I2C0 SCL
26 | GP20 | I2C0 SDA
25 | GP19 | SPI0 TX | I2C1 SCL
24 | GP18 | SPI0 SCK | I2C1 SDA
23 | GND
22 | GP17 | SPI0 CSn | I2C0 SCL | UART0 RX
21 | GP16 | SPI0 RX | I2C0 SDA | UART0 TX
1
2
39
LED
USB
BOOTSEL
Raspberry Pi Pico ©2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO and PIO
ADC
SPI
I2C
System Control
Debugging

SOFTWARE USED

THONNY:

Thonny is an integrated development environment for Python that is designed for beginners. It was created by Aivar Annamaa, an Estonian programmer. It supports different ways of stepping through code, step-by-step expression evaluation, detailed visualization of the call stack and a mode for explaining the concepts of references and heap.

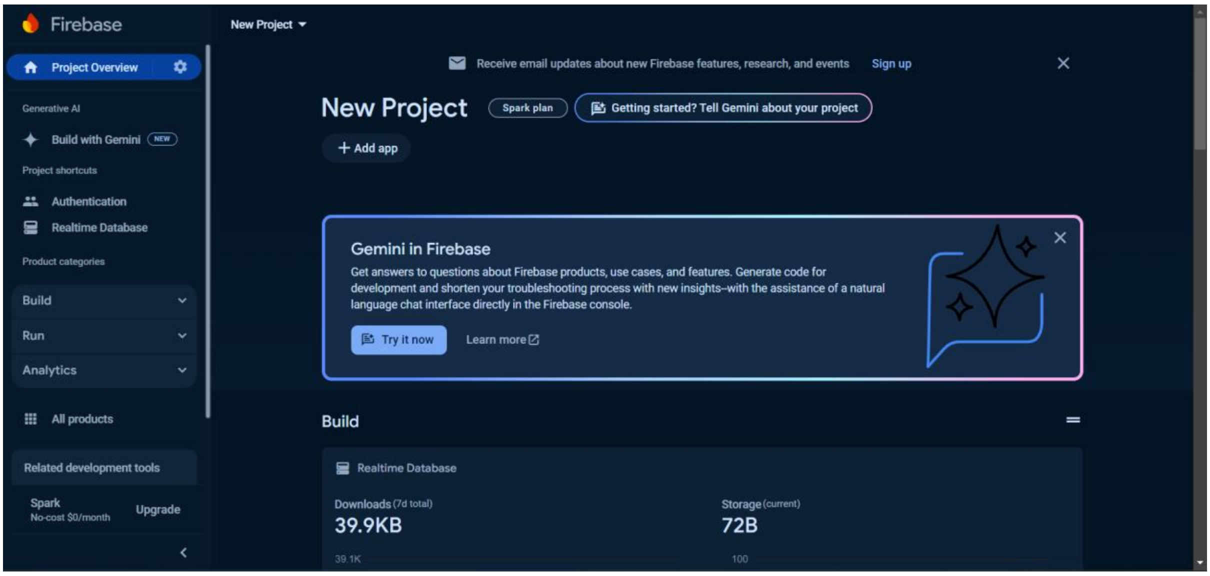
FIREBASE:

Firebase is a platform developed by Google for creating and managing mobile and web applications. It provides a comprehensive suite of tools and services to help developers build high-quality apps quickly and efficiently. Here are some key aspects of Firebase:

1. Realtime Database: A NoSQL cloud database that allows for real-time data synchronization across all clients. It supports offline capabilities and is ideal for applications requiring live data updates.
2. Firestore: A flexible, scalable database for mobile, web, and server development. It offers richer data model features, more advanced queries, and better integration with other Firebase services compared to the Realtime Database.
3. Authentication: Provides easy-to-use authentication solutions, supporting various sign-in methods including email/password, phone numbers, and federated identity providers like Google, Facebook, and Twitter.
4. Cloud Functions: Serverless functions that allow developers to run backend code in response to events triggered by Firebase features and HTTPS requests, without managing servers.
5. Cloud Storage: Secure and scalable file storage solution for storing and serving user-generated content such as images, audio, and videos.
6. Analytics: Google Analytics integration provides detailed insights into user behavior and app performance, allowing developers to make data-driven decisions.
7. Cloud Messaging: Firebase Cloud Messaging (FCM) enables sending notifications and messages to users across platforms (iOS, Android, and web) for engagement and updates.

Firebase is popular for its ease of use, scalability, and integration with other Google services, making it a valuable tool for developers looking to streamline app development and maintenance.

FIREBASE SETUP:



Receive email updates about new Firebase features, research, and events

New Project

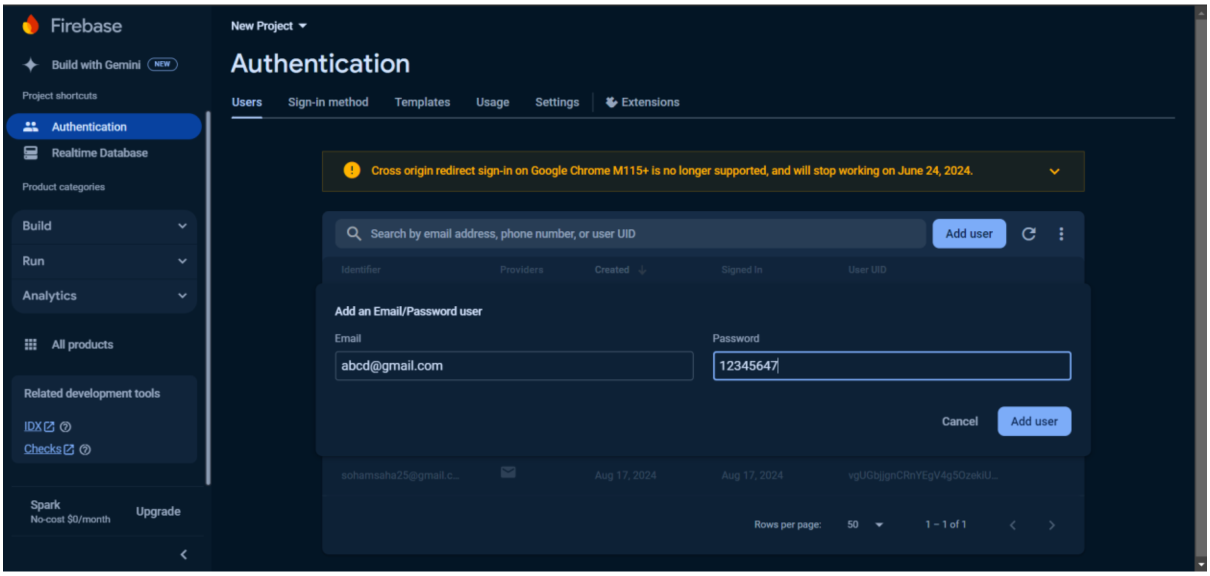
Getting started? Tell Gemini about your project

Gemini in Firebase

Get answers to questions about Firebase products, use cases, and features. Generate code for development and shorten your troubleshooting process with new insights–with the assistance of a natural language chat interface directly in the Firebase console.

Downloads (7d total) 39.9KB

Storage (current) 72B



Authentication

Cross origin redirect sign-in on Google Chrome M115+ is no longer supported, and will stop working on June 24, 2024.

Search by email address, phone number, or user UID

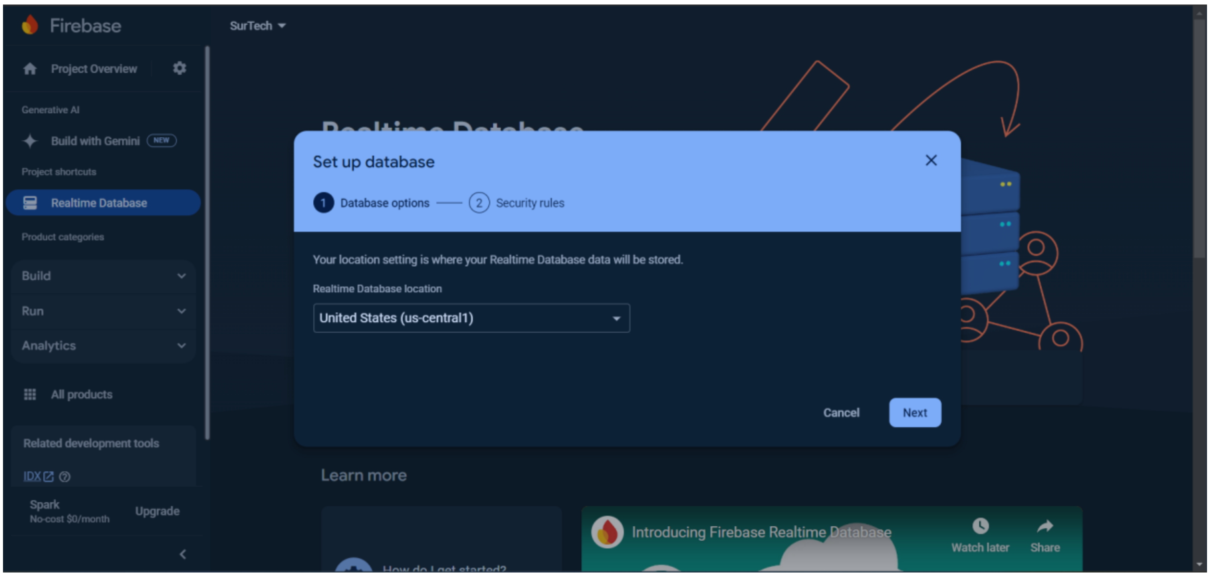
Add an Email/Password user

Email: abcd@gmail.com

Password: 12345647

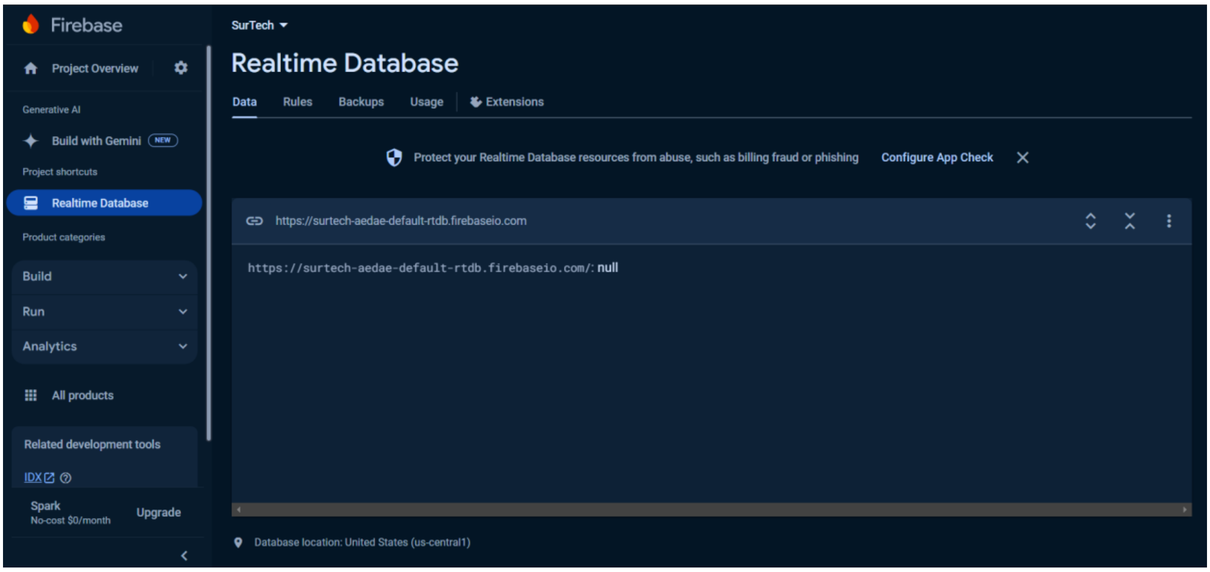
sohamsaha25@gmail.c... Aug 17, 2024 Aug 17, 2024 vgUGbjjgnCRnYEgV4g5OzekiU...

Rows per page: 50 1 – 1 of 1



Your location setting is where your Realtime Database data will be stored.

Realtime Database location: United States (us-central1)



Realtime Database

Protect your Realtime Database resources from abuse, such as billing fraud or phishing

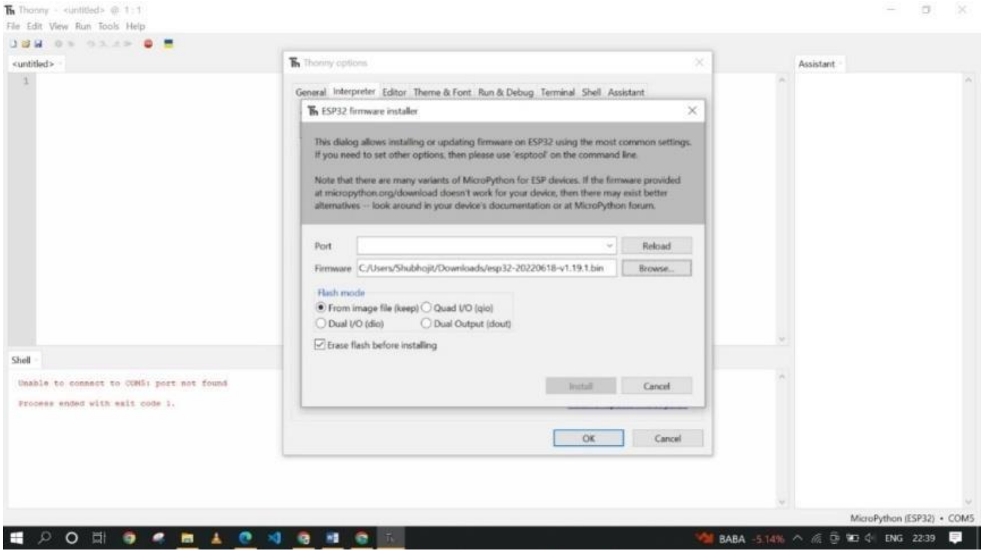
https://surtech-aedae-default-rtdb.firebaseio.com

```
https://surtech-aedae-default-rtdb.firebaseio.com/: null
```

Database location: United States (us-central1)

INSTALLATION OF MICROPYTHON USING THONNY IDE:

Step – 1:


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Shell
Assistant
Thonny options
General Interpreter Editor Theme & Font Run & Debug Terminal Shell Assistant
ESP32 firmware installer
Reload
Browse...
Install
Cancel
OK
Cancel
MicroPython (ESP32) • COM5
BABA -5.14% ENG 22:39

This dialog allows installing or updating firmware on ESP32 using the most common settings. If you need to set other options, then please use 'esptool' on the command line.

Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download doesn't work for your device, then there may exist better alternatives -- look around in your device's documentation or at MicroPython forum.

Port:

Firmware: C:/Users/Shubhojit/Downloads/esp32-20220618-v1.19.1.bin

Flash mode

● From image file (keep) ○ Quad I/O (qio)

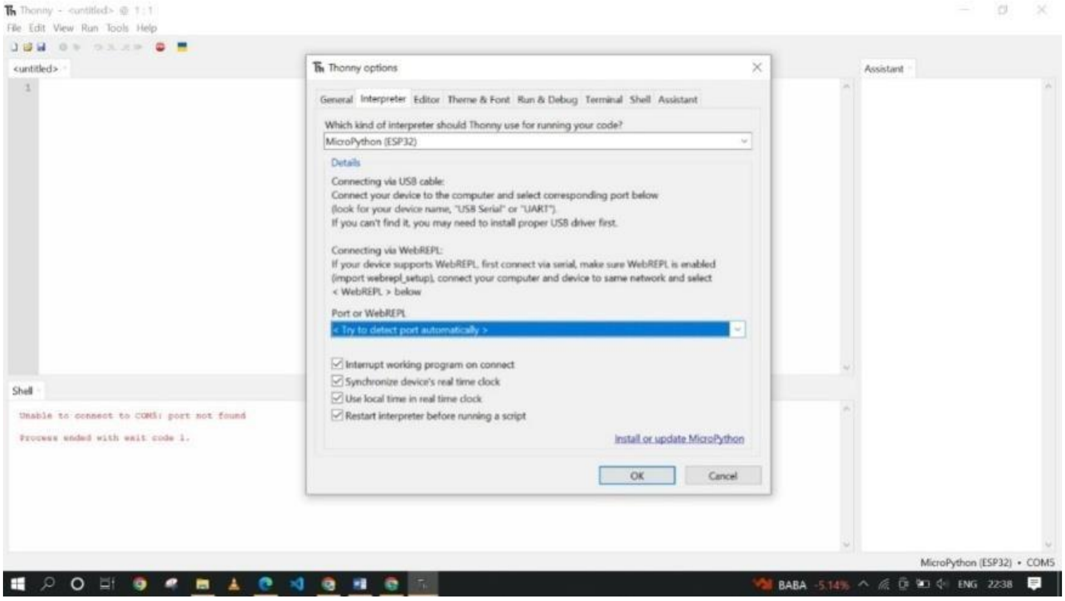
○ Dual I/O (dio) ○ Dual Output (dout)

☑ Erase flash before installing

```
Unable to connect to COM5: port not found

Process ended with exit code 1.
```

Step – 2:



Which kind of interpreter should Thonny use for running your code?
MicroPython (ESP32)

Connecting via USB cable:
Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Connecting via WebREPL:
If your device supports WebREPL, first connect via serial, make sure WebREPL is enabled
(import webrepl_setup), connect your computer and device to same network and select
< WebREPL > below

Port or WebREPL
< Try to detect port automatically >

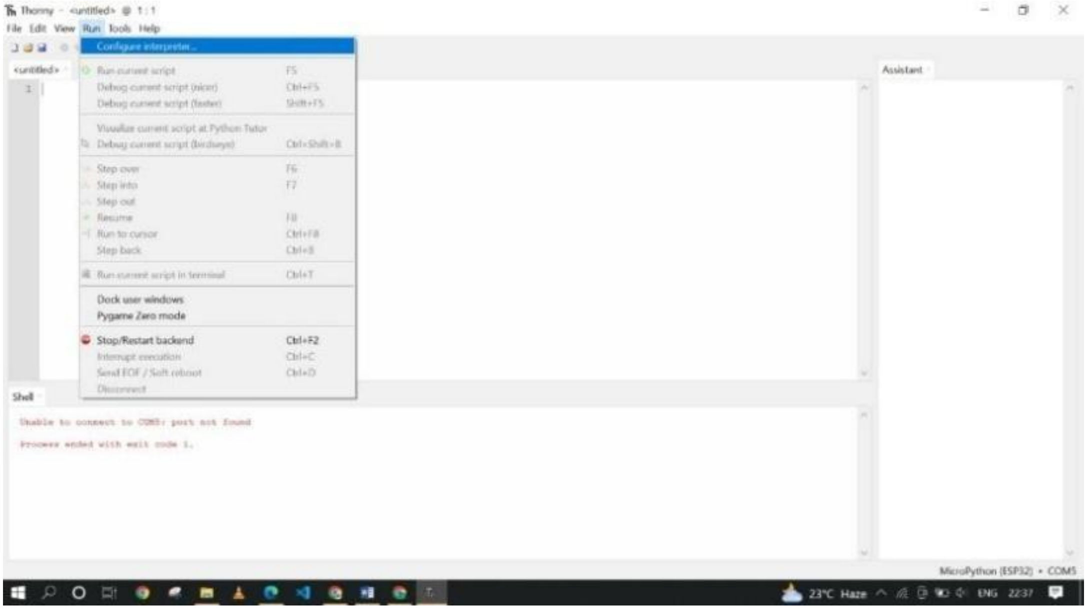
☑ Interrupt working program on connect
☑ Synchronize device's real time clock
☑ Use local time in real time clock
☑ Restart interpreter before running a script

```
Unable to connect to COM5: port not found

Process ended with exit code 1.
```

MicroPython (ESP32) • COM5

Step – 3:


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Configure interpreter...
Run current script F5
Debug current script (nicer) Ctrl+F5
Debug current script (faster) Shift+F5
Visualize current script at Python Tutor
Debug current script (birdseye) Ctrl+Shift+B
Step over F6
Step into F7
Step out
Resume F8
Run to cursor Ctrl+F8
Step back Ctrl+B
Run current script in terminal Ctrl+T
Dock user windows
Pygame Zero mode
Stop/Restart backend Ctrl+F2
Interrupt execution Ctrl+C
Send EOF / Soft reboot Ctrl+D
Disconnect
Shell
Assistant
23°C Haze
ENG
22:37

```
Unable to connect to COM5: port not found

Process ended with exit code 1.
```

MicroPython (ESP32) • COM5

ULTRASONIC SENSOR:

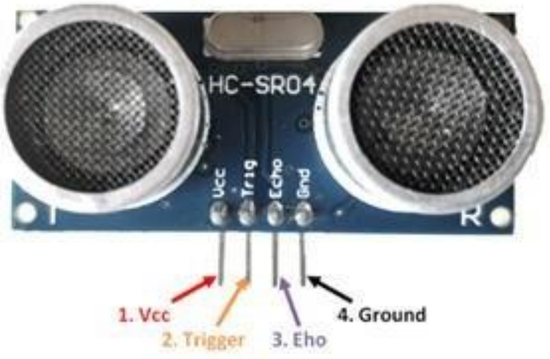
In this project we use **HC SR04** ultrasonic sensor.

HOW TO USE:

- First connect the pins of sensor accordingly with jumper wire with the Raspberry Pi Pico W.
- Then placed it to the top to the container to measure the whole length of the container.
- Then power on the Raspberry Pi Pico W and run proper code to generate sound wave from TRIGGER of the sensor.
- Then the wave hits the floor of container/goods keep in container and reverse back to ECHO.
- Now calculating the distance travelled by the wave to measure how much the container empty anddone.

HC SR04 PINOUT:

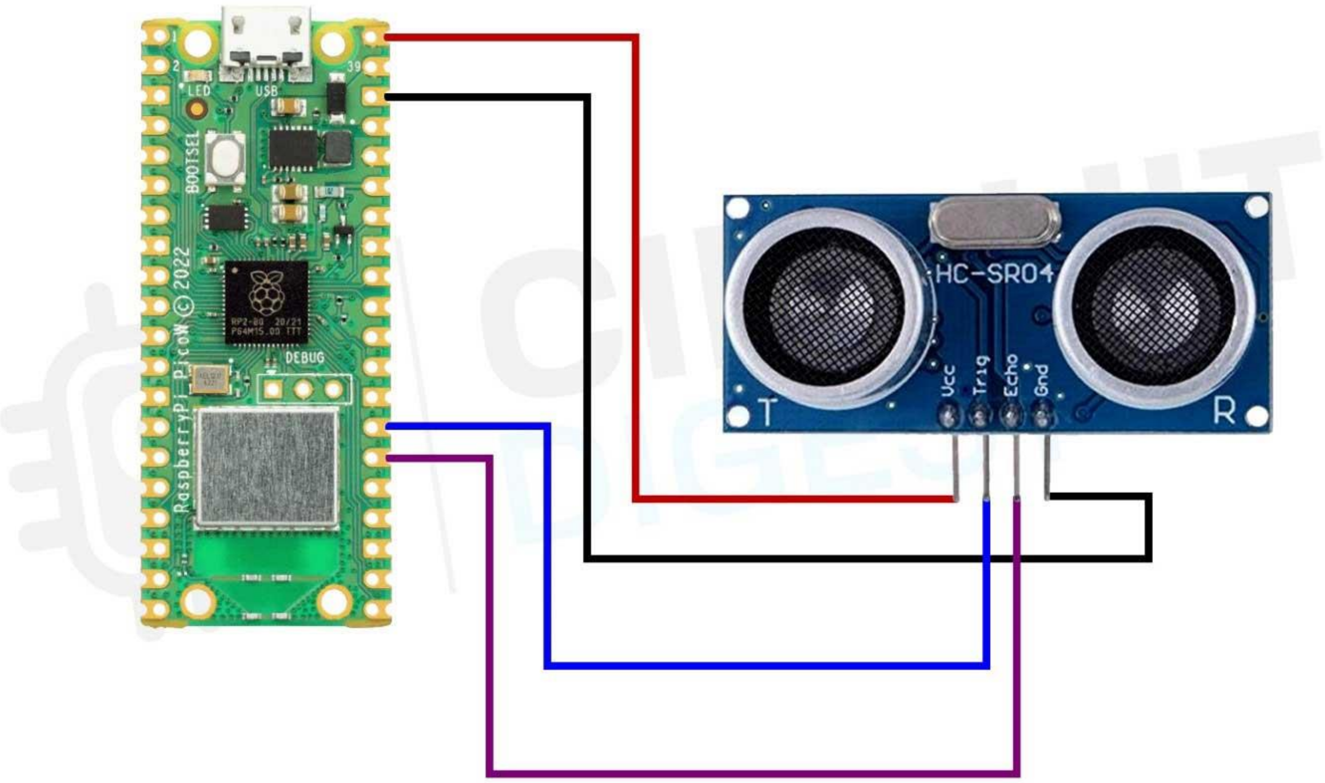
Vcc: Vcc is the pin for giving power to the sensor. It is connected to Vin pin of the ESP-32 as it is required 5v tooperate this.
Trigger: Trigger is the pin to generate a high and low pulse. As it is a digital sensor so trigger pin is connected to adigital output pin of ESP-32.
Echo: Echo is the pin to capture reflecting wave. As it is a digital sensor so echo pin is connected to a digital outputpin of ESP-32.
Gnd: Gnd is ground pin and connected to ground pin output of the ESP-32.



HOW HC SR04 WORK:

Basically the sensor generates soundwaves which goes to the targeted surface and reverse back to it. Thus it showsthe time taken by the pulse to reverse back. We know the velocity of sound in air and here the pulse travel the samedistance two times. So first we multiply the time with velocity of sound in air and then divide the result by two in the code. Hence the distance is calculated with this sensor. It is able to measure distance from 02 cm to 400cm.

So we can use this concept on really large container containing more essential stuffs and may implement it to largescale.



CONCLUSION

The system as the name indicates, “Smart Water Management System Using IoT” makes the system more flexible and provides an attractive user interface compared to other home automation systems. The proposed system can provide the data of water utilisation of each consumer on daily basis. A novel architecture for smart water management system is proposed using relatively new technologies. The system consists of mainly two components: an Raspberry Pi Pico W board, and HC SR04 ultrasonic sensor. We hide the complexity of the notions involved in the Smart Water Management System by including them into a simple, but comprehensive set of related concepts. This simplification isneeded to fit as much of the functionality on the limited space offered by a smart device’s display. This paper proposes a low cost, secure, ubiquitously accessible, auto-configurable, remotely controlled solution.Instead of fixed water tax, one can pay the charges as per utilisation of water. The system may also be useful in spreading the awareness of proper utilisation of water. Authorities can define the rate of water according to utilisation pattern and people may use the water

The system design and architecture were discussed, and the prototype presents the basic level of home appliance control and remote monitoring has been implemented. Finally, the proposed system is relatively new and better from the scalability and flexibility point of view than the commercially available storage systems.