Mini Industrial Project Report

SMART ACCIDENT DETECTION SYSTEM USING IOT

Submitted in partial fulfilment of the employment enhancementprogram of IT department

By

Avilash Bhowmick (14200221004)

Debjit Das (14200221002)

Soumyadeep Das (14200221021)

Sayan Surai (14200221017)

Mohor Banerjee (14200221042)

Third year

Meghnad Saha Institute of Technology

Under the supervision of

Mr. Sk Suman

IOT and Android DeveloperAcademy

Academy Of Skill Development

DEPARTMENT OF INFORMATION TECHNOLOGY

I hereby submit the project report prepared by me under the supervision of Mr. Sk Suman entitled, "SMART ACCIDENT DETECTION USING IOT**"** for the partial fulfilment of the employment enhancement program of IT department of Meghnad Saha Institute of Technology, affiliated to Maulana Abul Kalam Azad University of Technology (MAKAUT).

_____

Mr. Sk Suman

IOT and Android DeveloperAcademy Of

Skill Development.

_____ _____

Prof. Subir Hazra
HOD
IT, MSIT

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful. We take this opportunity to express our deep gratitude towards our project mentor, **Mr. SK Suman** for giving such valuable suggestions, guidance and encouragement during the development of this project work. Last but not the least we are grateful to all the faculty members of Academy Of Skill Development for their support.

ABSTRACT

The “Smart Car Accident System” is an innovative Internet of Things (IoT) solution meticulously designed to enhance vehicle safety. This system ingeniously integrates a variety of hardware components. The Ultra Sound - HCSR04 is used for precise distance measurement, while the GPS - ublox neo6m provides accurate geolocation data.

The Gyro - MPU6050 measures rotational motion, and the Servo Motor - SG90 controls mechanical movement. The system also includes a Buzzer for audible alerts, an LED for visual indicators, and an ESP32 microcontroller with Wi-Fi capabilities.

The system is powered by a 5-volt DC Power Supply and uses a Wi-Fi Module for wireless communication. The software integration is achieved using Thonny version v3.1.0 and Micropython v1.22.1 with a generic ESP32 module.

A user-friendly application, developed in Android Studio, provides features like live location tracking, path coverage, and accident notifications. All backend services are hosted on Google Firebase, ensuring reliable and secure data management. This project is a testament to the potential of IoT in revolutionizing vehicle safety.

CONTENTS:

INTRODUCTION

In the realm of Internet of Things (IoT), a groundbreaking project has been undertaken, christened as the “Smart Car Accident System”. This system is a testament to the transformative power of technology, specifically IoT, in enhancing vehicle safety.

The system is a harmonious blend of various hardware components, each serving a unique purpose. The HCSR04 Ultrasonic sensor measures distance, the ublox neo6m GPS module provides geolocation data, and the MPU6050 Gyro sensor measures rotational motion.

The SG90 Servo Motor controls mechanical movements, while the Buzzer and LED provide audio-visual alerts. The heart of the system is the ESP32 microcontroller, which, along with the Wi-Fi Module, enables wireless communication. All these components are powered by a 5-volt DC Power Supply and connected using Jumpers.

On the software front, the system leverages Thonny and Micropython for code integration with the ESP32. A user-friendly application, developed in Android Studio, serves as the interface for users to interact with the system. The application offers features like live location tracking, path coverage, and accident notifications.

Google Firebase hosts all backend services, ensuring reliable and secure data management.

The “Smart Car Accident System” is not just a project; it’s a step towards a safer future. It exemplifies how IoT can be harnessed to create solutions that have a real-world impact, making our roads safer and saving precious lives. This project is a beacon of innovation, illuminating the path for future IoT solutions in vehicle safety.

COMPONENTS USED

Hardware Components :

- **Ultra Sound - HCSR04**
- **GPS - ublox neo6m**
- **Gyro - MPU6050**
- **Servo Motor - SG90**
- **Buzzer**
- **LED**
- **ESP32**
- **Jumper**
- **5-volt DC Power Supply**
- **Wi-Fi Module**

DESCRIPTION

- **ESP - 32**

In the dynamic landscape of Internet of Things (IoT), a pioneering project has emerged, known as the “Smart Car Accident System”. This system is a manifestation of the transformative potential of IoT in augmenting vehicle safety.

The heart of the system is the ESP32, a low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. Developed by Espressif Systems, a Shanghai-based Chinese company, and manufactured by TSMC using their 40 nm process, the ESP32 is a successor to the ESP8266 microcontroller. It employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor.

It includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules.

The system harmoniously integrates various hardware components, each serving a unique purpose. The HCSR04 Ultrasonic sensor measures distance, the ublox neo6m GPS module provides geolocation data, and the MPU6050 Gyro sensor measures rotational motion. The SG90 Servo Motor controls mechanical movements, while the Buzzer and LED provide audio-visual alerts.

On the software front, the system leverages Thonny and Micropython for code integration with the ESP32. A user-friendly application, developed in Android Studio, serves as the interface for users to interact with the system. The application offers features like live location tracking, path coverage, and accident notifications. Google Firebase hosts all backend services, ensuring reliable and secure data management.

The “Smart Car Accident System” is more than just a project; it’s a stride towards a safer future. It exemplifies how IoT can be harnessed to create solutions that have a real-world impact, making our roads safer and saving precious lives. This project is a beacon of innovation, illuminating the path for future IoT solutions in vehicle safety.

Features of the ESP-32:

- Single or Dual-Core 32-bit LX6 Microprocessor with clock frequency up to 240 MHz.
- 520 KB of SRAM, 448 KB of ROM and 16 KB of RTC SRAM.
- Supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps.
- Support for both Classic Bluetooth v4.2 and BLE specifications.
- 34 Programmable GPIOs.
- Up to 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity include 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
- Ethernet MAC for physical LAN Communication (requires external PHY).
- 1 Host controller for SD/SDIO/MMC and 1 Slave controller for SDIO/SPI.
- Motor PWM and up to 16-channels of LED PWM.
- Secure Boot and Flash Encryption.
- Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC and RNG.

ESP-32 HARDWARE PART:

Espressif Systems released several modules based on ESP32 and one of the popular options is the ESP-WROOM-32 Module. It consists of ESP32 SoC, a 40 MHz crystal oscillator, 4 MB Flash IC and some passive components.

The good thing about ESP-WROOM-32 Module is the PCB has edge castellation. So, what third-part manufacturers do is take the ESP-WROOM-32 Module and design a break-out board for this module.

One such board is the ESP32 DevKit Board. It contains the ESP-WROOM-32 as the main module and also some additional hardware to easily program ESP32 and make connections with the GPIO Pins.

There's also 520 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IoT devices nowadays.

ESP32 implements TCP/IP, full 802.11 b/g/n/e/i WLAN MAC protocol, and Wi-Fi Direct specification. This means ESP32 can speak to most of the WiFi Routers out there when used in station(client) mode. Also it is able to create an Access point with full 802.11 b/g/n/e/i.

Power Requirement

The operating voltage of ESP32 ranges from 2.3 V to 3.6 V. When using a single-power supply, the recommended voltage of the power supply is 3.3 V, and its recommended output current is 500 mA or more.

- PSRAM and flash both are powered by VDD_SDIO. If the chip has an embedded flash, the voltage of
- VDD_SDIO is determined by the operating voltage of the embedded flash. If the chip also connects to an
- external PSRAM, the operating voltage of external PSRAM must match that of the embedded flash. This
- also applies if the chip has an embedded PSRAM but also connects to an external flash.
- When VDD_SDIO 1.8 V is used as the power supply for external flash/PSRAM, a 2 kΩ grounding resistor
- should be added to VDD_SDIO. For the circuit design, please refer to Figure ESP32•WROVER
- Schematics, in ESP32-WROVER Datasheet.
- When the three digital power supplies are used to drive peripherals, e.g., 3.3 V flash, they should comply with the peripherals' specifications.

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

Power modes:

- **Active mode:** The chip radio is powered on. The chip can receive, transmit, or listen.
- **Modem sleep mode:** The CPU is operational and the clock is configurable. The Wi-Fi/Bluetoothbaseband and radio are disabled.
- **Lightsleep mode**: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP coprocessors are running. Any wake-up events (MAC, host, RTC timer, or external interrupts)will wake up the chip.
- **Deep•sleep mode**: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP coprocessor is functional.
- **Hibernation mode**: The internal 8 MHz oscillator and ULP coprocessor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and certain RTC GPIOs are active. TheRTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

- **Peripherals and I/O**

1. GPIO (General Purpose Input/Output)

The ESP32 microcontroller used in this project has several GPIO pins. These pins can be programmed as input or output and used for a variety of purposes. In this project, they are used to interface with various components such as the ultrasonic sensor, servo motor, buzzer, and LEDs.

2. ADC (Analog to Digital Converter)

While not explicitly mentioned in your project, the ESP32 does have built-in ADCs. An ADC converts analog signals into digital values, which can be useful when dealing with sensors that provide analog outputs.

3. DAC (Digital to Analog Converter)

Similarly, a DAC converts digital values into analog signals. This can be useful for controlling devices that require an analog input.

4. Ultrasonic Sensor (HCSR04)

The HCSR04 ultrasonic sensor is used to measure distance. It sends out an ultrasonic wave, and by measuring the time it takes for the wave to return after bouncing off an object, it can calculate the distance to that object. This feature is used in two ways in your project:

To detect an accident: If the distance to an object suddenly decreases, it could indicate a collision.
For parking assistance: The sensor can help determine the distance to obstacles while parking, and the LEDs blink accordingly.

5. Servo Motor (SG90)

The SG90 servo motor is used to control the deployment of the airbag in case of an accident. If the MPU6050 gyro sensor detects a change in the normal states of the x, y, z axes indicating a potential accident, the servo motor is activated to deploy the airbag.

6. Buzzer

The buzzer is used to provide audible alerts in certain situations, such as when an accident is detected or when the vehicle is too close to an object while parking.

7. LEDs

LEDs are used to provide visual indicators. For example, they can blink to indicate the distance to obstacles while parking, with the frequency of blinking increasing as the vehicle gets closer to the obstacle.

This project is a great example of how various components can work together in an IoT system to provide a useful and potentially life-saving service.

8. I2C Interface

ESP32 has two I2C bus interfaces which can serve as I2C master or slave, depending on the user's

configuration. The I2C interfaces support:

• Standard mode (100 Kbit/s)

• Fast mode (400 Kbit/s)

• Up to 5 MHz, yet constrained by SDA pull-up strength• 7-bit/10-bit addressing mode

• Dual addressing mode

Users can program command registers to control I2C interfaces, so that they have more flexibility

9. I2S Interface

Two standard I2S interfaces are available in ESP32. They can be operated in master or slave mode, in full duplex and half-duplex communication modes, and can be configured to operate with an 8-/16-/32-/48-/64-bit resolution as input or output channels. BCK clock frequency, from 10 kHz up to 40 MHz, is supported.

Whenone or both of the I2S interfaces are configured in the master mode, the master clock can be output to the external DAC/CODEC.

Both of the I2S interfaces have dedicated DMA controllers. PDM and BT PCM interfaces are supported.

- **On-board Switches & LED Indicator**

EN

Reset button: pressing this button resets the system.

Boot

Download button: holding down the Boot button and pressing the EN button initiates the firmware download mode.Then user can download firmware through the serial port.

- **Switches & Indicators**

The board also has a LED indicator which shows that the board has power.

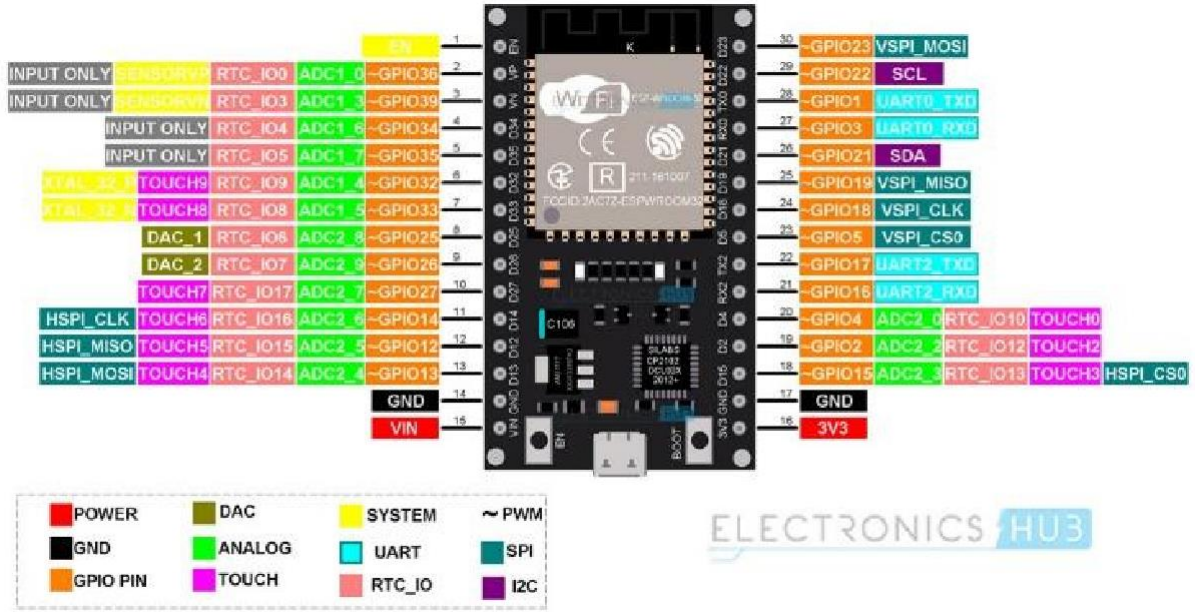
- **Serial Communication**

The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USB signal to serialand allows your computer to program and communicate with the ESP32 chip.

- CP2102 USB-to-UART converter
- 4.5 Mbps communication speed

If you have an older version of CP2102 driver installed on your PC, we recommend upgrading now.

- **ESP – 32 Pinout**



GPIO:

The most commonly used peripheral is the GPIO. ESP32 has 34 GPIO pins with each pin carrying out more than onefunction (only one will be active). You can configure a pin as either a GPIO or an ADC or an UART in the program.

ADC and DAC pins are predefined and you have to use the manufacturer specified pins. But other functions likePWM, SPI, UART, I2C etc. can be assigned to any GPIO pin through program.

RTC GPIO:

ESP32 has 16 RTC GPIOs, which are part of the RTC Low-Power subsystem. These pins can be used to wake ESP32from deep sleep as external wake-up source.

ADC:

ESP32 has two 12-bit SAR Analog to Digital Converter Modules with 8-channels and 10-channels each. So, ADC1 andADC2 blocks combined together have 18 channels of 12-bit ADC.

With 12-bit resolution, the output Digital values will be in the range of 0 – 4093.

DAC:

ESP32 Microcontroller has two independent 8-bit Digital to Analog Converter channels to convert digital values to analog voltage signals. The DAC has internal resistor network and uses power supply as input reference voltage.

The following two GPIO Pins are associated with DAC functionalities.

DAC1 —

GPIO25DAC2

— GPIO26

Capacitive Touch GPIOs

The ESP32 SoC has 10 capacitive-sensing GPIOs, which can detect variations in capacitance on a pin due to touchingor approaching the GPIO Pin with a finger or stylus. These Touch GPIOs can be used in implementing capacitive touch pads, without any additional hardware.

SPI:

The ESP32 Wi-Fi chip features three SPI blocks (SPI, HSPI and VSPI) in both master and slave modes. SPI is used tointerface with Flash Memory. So, you have two SPI interfaces.

I2C:

There are two I2C interfaces in ESP32 with complete flexibility on assigning pins i.e., SCL and SDA pins for both I2C interfaces can be assigned in the program by the user.

If you are using Arduino IDE, then the default I2C pins are:

SDA –

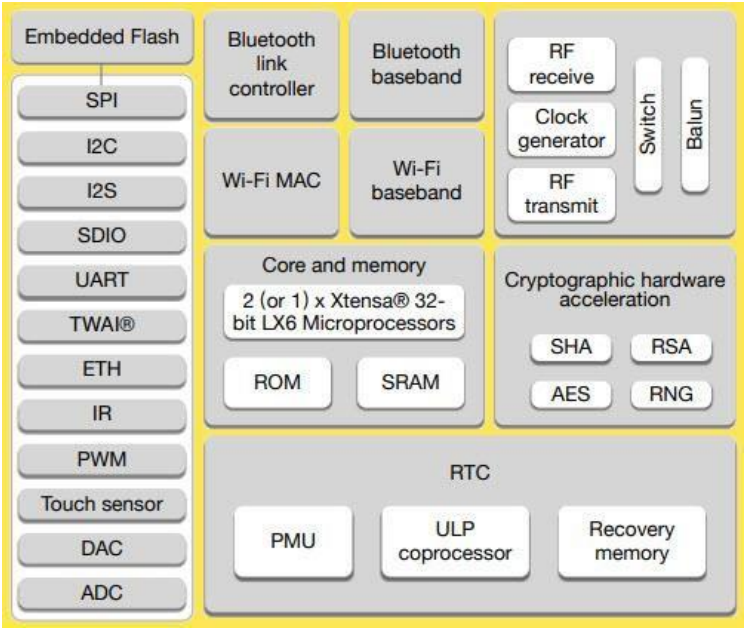
GPIO21SCL

– GPIO22

PWM:

The PWM Controller in ESP32 have 16 independent PWM waveform channels with configurable frequency and duty cycle. The PWM waveform can be used to drive motors and LEDs. You can configure the PWM signal frequency, channel, GPIO pin and also the duty cycle.

- **ESP-32 ARCHITECTURE:**



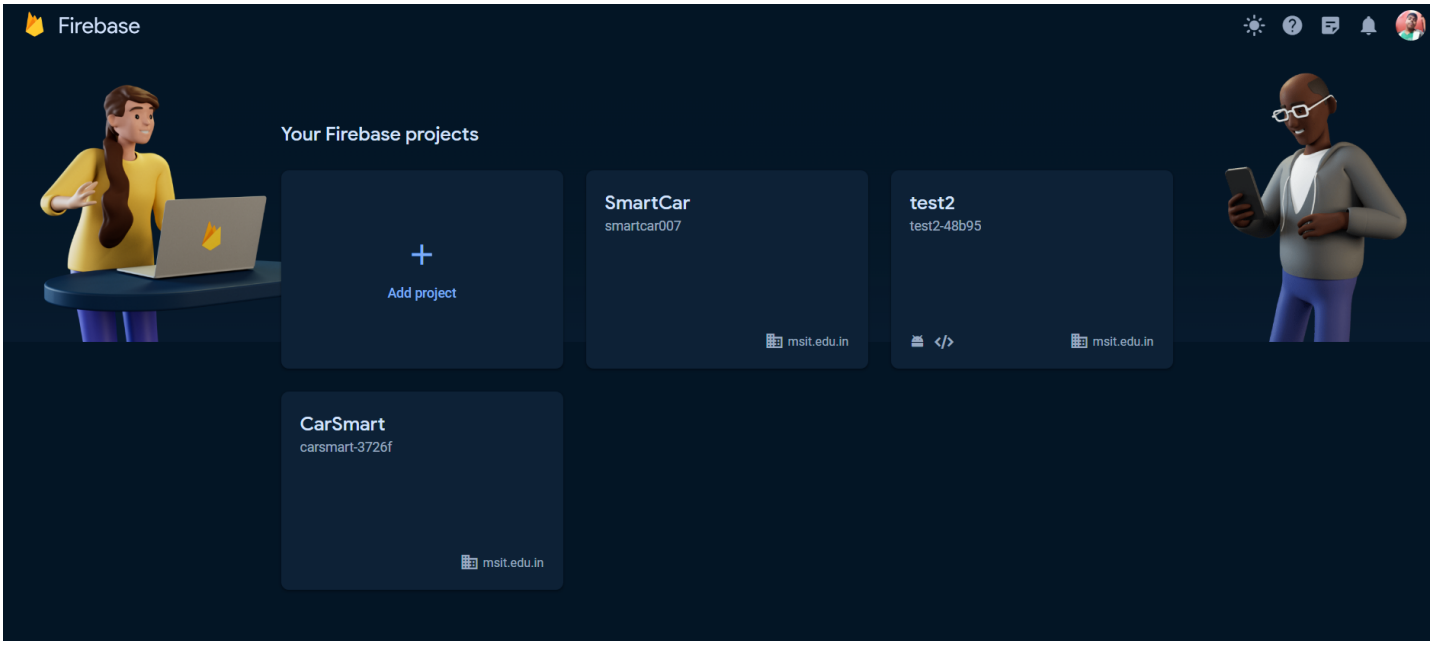
SOFTWARE USED:

- ## FIREBASE:

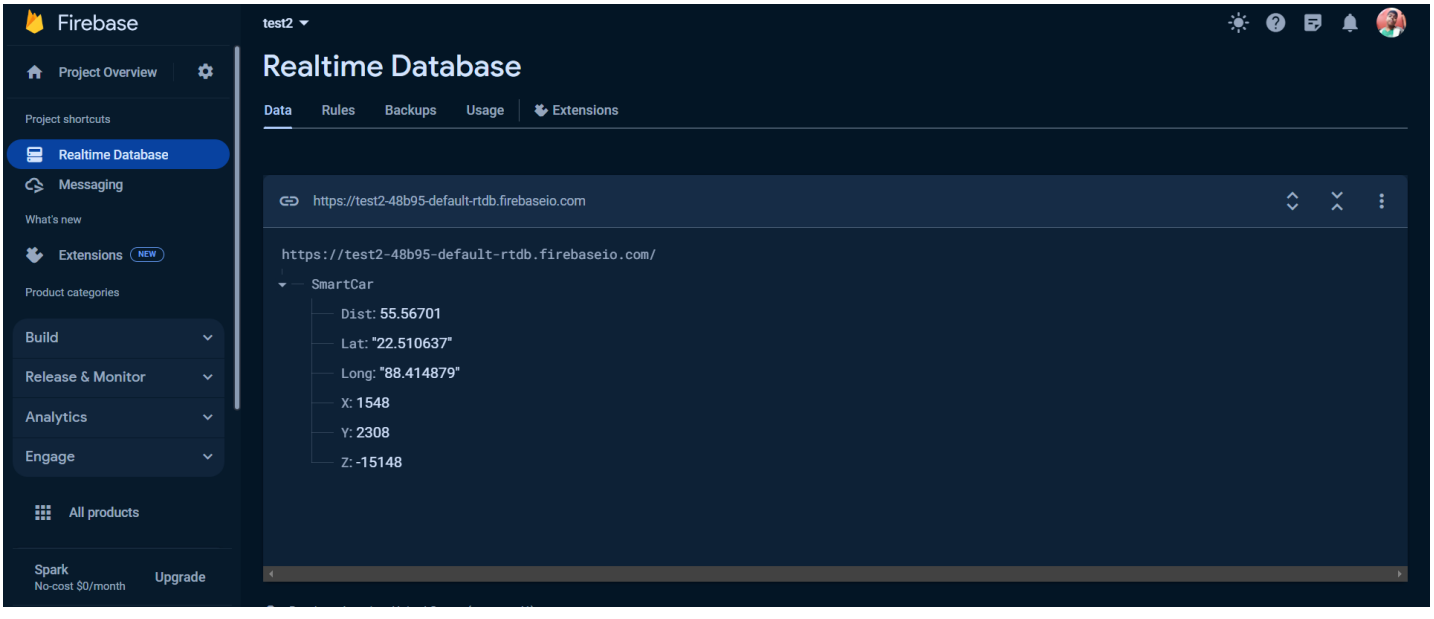
Firebase is a Google-backed platform that provides a range of products and solutions for every stage of your app development journey, from backend infrastructure to analytics, engagement, and more. Firebase offers fully managed backend infrastructure to accelerate app development, including Cloud Firestore for database management and Authentication for user authentication. Firebase also provides solutions for release and monitoring, such as Crashlytics for crash reporting and Performance Monitoring for performance analysis. Firebase's analytics solutions include Google Analytics for app usage analysis, Remote Config for remote app configuration, and Cloud Messaging for push notifications. Firebase is trusted by many development teams around the world .

CONNECTING FIREBASE DATABASE :

Step – 1:



Step – 2:



Step – 3:

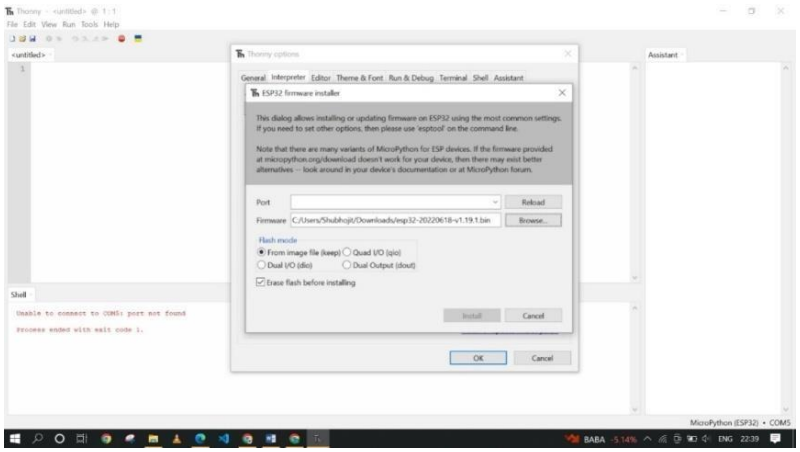
```
14
15 URL='test2-48b95-default-rtdb.firebaseio.com/SmartCar'
16
```

- **THONNY:**

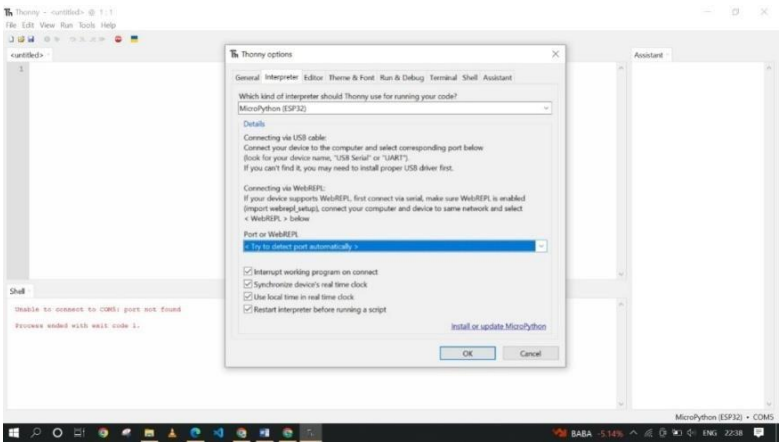
Thonny is an integrated development environment for Python that is designed for beginners. It was created by Aivar Annamaa, an Estonian programmer. It supports different ways of stepping through code, step-by-step expression evaluation, detailed visualization of the call stack and a mode for explaining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

INSTALLATION OF MICROPYTHON USING THONNY IDE:

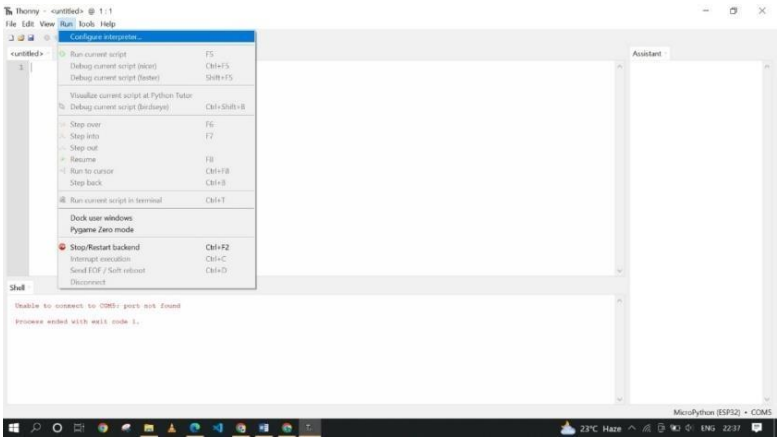
Step – 1:



Step – 2:



Step – 3:


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
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
<untitled>
Assistant
Shell
Unable to connect to COM5: port not found
Process ended with exit code 1.
MicroPython (ESP32) • COM5
23°C Haze
ENG 22:37

- **ULTRASONIC SENSOR:**

In this project we use **HC SR04** ultrasonic sensor. Here is a brief discussion about it-

HOW TO USE:

- First connect the pins of sensor accordingly with jumper wire with the ESP-32.
- Then placed it to the top to the container to measure the whole length of the container.
- Then power on the ESP-32 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 and done.

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 output pin of ESP-32.

Gnd: Gnd is ground pin and connected to ground pin output of the ESP-32.

HOW HC SR04 WORK:

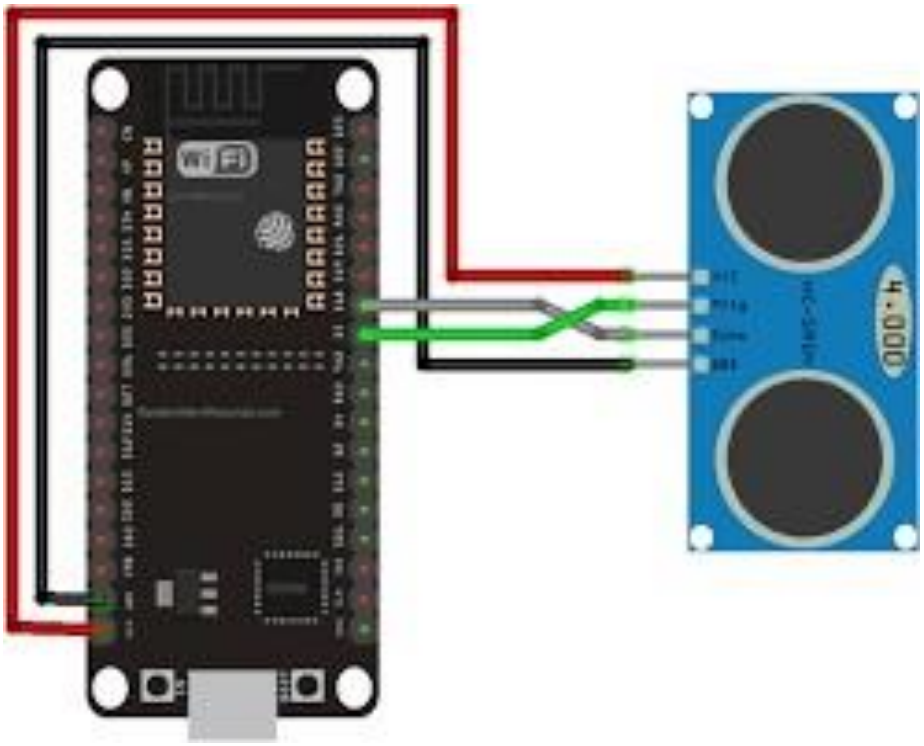
Basically the sensor generates soundwaves which goes to the targeted surface and reverse back to it. Thus it showsthe time taken by the pulse to reverse back. We know the velocity of sound in air and here the pulse travel the same distance two times. So first we multiply the time with velocity of sound 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.

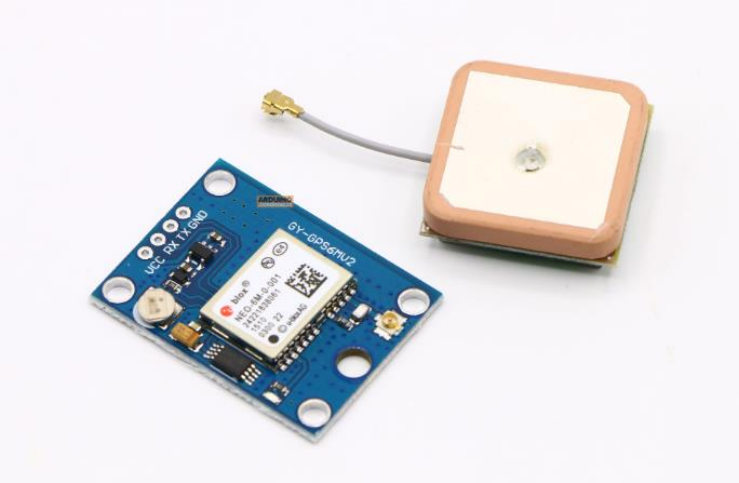
The following steps have done to **calibrate and implement** it in this project:

- First we connect the sensor with ESP-32 accordingly- Vcc to Vin, Trigger to D2, Echo to D5 and Gnd to Gnd.
- Then fit it with the above surface of the container with an angle so it can measure the distance accurately.
- Then upload the proper code to the ESP-32 and run it.
- As soon as we run the code, according to the code first the trigger pin will low for 10 microsecond to generate low pulse and high for 10 microsecond to generate high pulse and then again low.
- Then the echo pin will on and capture the pulse to measure time taken by the pulse to go and back to the sensor.
- Then we get the distance on the serial monitor of Thonny ide as well as Blynk cloud server.
- Then we measure the exact distance with ruler to ensure it's accuracy and as expected two distances are equal.

CIRCUIT DIAGRAM :



- **GPS- ublox neo6m**



The u-blox NEO-6M is a GPS (Global Positioning System) module manufactured by u-blox. It is commonly used in various applications where accurate positioning information is required. Here are some key features and information about the u-blox NEO-6M GPS module:

1. GPS Receiver: The NEO-6M is a standalone GPS receiver module, meaning it can receive signals form GPS satellites and determine its position without the need for external processing.
2. U-blox Technology: u-blox is a Swiss company known for its positioning and wireless communication technologies. The NEO-6M uses u-blox's GPS technology to provide accurate and reliable positioning information.
3. Communication Interface: The module typically communicates with external devices (such as microcontrollers or computers) via a serial communication interface. It uses the NMEA(National Marine Electronics Association) protocol to send standard GPS sentences with position, velocity, and time information.
4. Serial Communication: The NEO-6M often features a UART (Universal Asynchronous Receiver-Transmitter) serial interface, making it compatible with a wide range of microcontrollers and communication modules.
5. Voltage Requirements: The module usually operates on low voltage, typically around 3.3V, making it suitable for integration into various for integration into various electronic projects.
6. Antenna Connection: The module requires an external GPS antenna for proper signal reception. The antenna is typically connected to the module through an SMA or U.FL connector.
7. Configuration: The NEO-6M can be configured using a configuration software provided by u-blox. This allows users to customize the module's behavior and optimize it for specific applications.
8. Cold Start and Warm Start: The NEO-6M supports cold start and warm start features. A cold start occurs when the module is powered on with no stored satellite information, while a warm start occurs when the module has some prior satellite information.
9. Accuracy and Sensitivity: The module provides position accuracy and sensitivity that is suitable for a wide range of applications, from navigation in vehicles to outdoor tracking.

Before using the u-blox NEO-6M, it's essential to refer to the module's datasheet and technical documentation provided by u-blox to understand its specifications, pinouts and configuration options. Additionally, ensure that we have a clear line of sight to GPS satellites for optimal performance.

The operating voltage of NEO-6M chip is from 2.7 to 3.6V. However, the NEO-6M module description says that it's "5V tolerant", but it's mostly because of the built-in 5V-3.3V voltage regulator, I think. I have never seen a NEO-6M GPS module that executes the requisite UART logic level shifting, but the module worked fine without a level-shifter while I was testing it not only with a PC but also with an Arduino (I'll be back here later).

It's cheerfully noted that NEO-6M-0-001 GPS receiver can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity while absorbing only 45mA supply current (Just around 11mA in Power Save Mode -PSM). It can also do up to 5 location updates in a second with 2.5m horizontal position accuracy, and the positioning engine boasts a Time-To- First-Fix (TTFF) of under 1 second. Since the backup battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1 second. This allows much faster position locks because without the battery the GPS always cold-start so the initial GPS lock takes much more time. The EEPROM chip together with the backup battery helps retain the battery backed RAM (BBR) that holds clock data, latest position data (GNSS orbit data) and module configuration. The battery is automatically recharged when power is applied and keeps data intact for up to two weeks without power.

- **Gyro - MPU6050**

Motion interface is rapidly becoming a key function in many consumer electronics devices including smartphones, tablets, gaming consoles, and smart-TVs as it provides an intuitive way for consumers to interact with electronic devices by tracking motion in free space and delivering these motions as input commands.The MPU-6050 is the world's first and only 6-axis MotionTracking devices designed for the low power, low cost, and high performance requirements of smartphones, tablets and wearable sensors.By combining a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die together with an onboard Digital Motion Processor(DMP) capable of processing complex 9-axis MotionFusion algorithms. The parts' integrated 9-axis MotionFusion algorithms access external magnetometers or other sensors through an auxiliary master I2C bus, allowing the devices to gather a full set of sensor data without intervention from the system processor.For precision tracking of both fast and slow motions, the parts feature a user-programmable gyro full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g

---

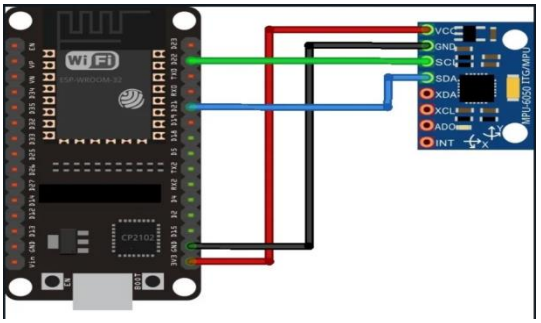
Features

- I2C Digital-output of 6 or 9-axis MotionFusion data in rotation matrix, quaternion, Euler Angle, or raw data format
- Input Voltage: 2.3 - 3.4V
- Selectable Solder Jumpers on CLK, FSYNC and AD0
- Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps
- Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g
- Digital Motion Processing™ (DMP™) engine offloads complex MotionFusion, sensor timing synchronization and gesture detection

- Embedded algorithms for run-time bias and compass calibration. No user intervention required
- Digital-output temperature sensor

Application

- BlurFree technology (for Video/Still Image Stabilization)
- AirSign technology (for Security/Authentication)
- TouchAnywhere technology (for “no touch” UI Application Control/Navigation)
- MotionCommand technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture iG gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys



- **Servo Motor - SG90**

SG90 is a popular micro servo motor commonly used in hobbyist and DIY projects. It is small, low cost servo motor that can rotate 180 degrees with a maximum torque of 1.8 kg-cm. It operates at 4.8-6V and has a weight of approximately 9 grams, making it ideal for small-scale robotics and model control applications. The SG90 is a digital servo motor. It uses some pulse width modulation signal to control its position and speed, making it all precise and accurate and stuff which makes it pretty suitable for all kinds of fancy applications, like robotics, and RC vehicles and all sorts of hobbyist projects.

The function of the servo motor is to convert the control signal of the controller into the rotational angular displacement or angular velocity of the motor output shaft. Servo motor is used to drive the joints.

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analog or digital) representing the position commanded for the output shaft.

The motor is paired with some type of position encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero, and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

More sophisticated servomotors make use of an Absolute encoder (a type of rotary encoder) to calculate the shafts position and infer the speed of the output shaft.[3] A variable-speed drive is used to control the motor speed.[4] Both of

these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

The first servomotors were developed with synchros as their encoders. Much work was done with these systems in the development of radar and anti-aircraft artillery during World War II.

Simple servomotors may use resistive potentiometers as their position encoder. These are only used at the very simplest and cheapest level and are in close competition with stepper motors. They suffer from wear and electrical noise in the potentiometer track. Although it would be possible to electrically differentiate their position signal to obtain a speed signal, PID controllers that can make use of such a speed signal, generally warrant a more precise encoder.

Modern servomotors use rotary encoders, either absolute or incremental. Absolute encoders can determine their position at power-on but are more complicated and expensive. Incremental encoders are simpler, cheaper, and work at faster speeds. Incremental systems, like stepper motors, often combine their inherent ability to measure intervals of rotation with a simple zero-position sensor to set their position at start-up.

Instead of servomotors, sometimes a motor with a separate, external linear encoder is used. These motor + linear encoder systems avoid inaccuracies in the drivetrain between the motor and linear carriage, but their design is made more complicated as they are no longer a pre-packaged factory-made system.

- **CODE:**

```
import con #connectivity
import test
import ufirebase as firebase #firebase connection
from machine import I2C, Pin
import mpu6050 # accident detection
import gps_sat # gps value
from time import sleep
from hcsr04 import HCSR04 # parking
from servo import Servo # airbag

test.res()

con.connection()

URL='test2-48b95-default-rtdb.firebaseio.com/SmartCar'

#Ultrasonic
sensor = HCSR04(trigger_pin=18,echo_pin=5,echo_timeout_us=1000000)

#ESP8266
# i2c = I2C(scl=Pin(5), sda=Pin(4))

#ESP32
i2c = I2C(scl=Pin(22), sda=Pin(21))

#Servo Motor

motor=Servo(pin=13)
motor.move(0)
accelerometer = mpu6050.accel(i2c)
Data={}
while True:
    val = accelerometer.get_values()
    gps_data = gps_sat.val2()

    parking_distance = sensor.distance_cm()
    if parking_distance <=7:
        Pin(12,Pin.OUT).value(1)
        Pin(14,Pin.OUT).value(1)
        Pin(27,Pin.OUT).value(1)
        print("Caution!!!! Obstacle is close")
    else:
        test.res()
    '''if (parking_distance >5 and parking_distance <=10):
        test.res()
        Pin(12,Pin.OUT).value(1)
        Pin(14,Pin.OUT).value(1)
    if (parking_distance >10 and parking_distance <=15):
        test.res()
        Pin(12,Pin.OUT).value(1)'''
```

```
    '''
    GyZ = val['GyZ']
    GyY = val['GyY']
    GyX = val['GyX']
    '''

    GyZ = val['AcZ']
    GyY = val['AcY']
    GyX = val['AcX']

    if(GyX <= -7500 or GyX >= 8500 or GyY <= -7500 or GyY >= 8500 or GyZ >=0):
        Pin(12,Pin.OUT).value(1)
        Pin(14,Pin.OUT).value(1)
        Pin(27,Pin.OUT).value(1)
        # Open Servo Motor
        motor.move(180)

        print("Accident Occured")
        print()
        print('X-axix',GyX)
        print('Y-axix',GyY)
        print('Z-axix',GyZ)
        # print("Gps data: ",lat,long)

        break
    else:
        motor.move(0)
    #Sending Accelerometer Data

    Data['X'] = GyX
    Data['Y'] = GyY
    Data['Z'] = GyZ

    lat = gps_data["Latitude"]
    long = gps_data["Longitude"]

    #Sending Lat , Lang Data

    Data['Lat'] = lat
    Data['Long'] = long

    Data['Dist'] = parking_distance

    #trying to send data
    while True:
        try:
            # print(Data)
            firebase.put(URL,Data) # Realtime
            #print('Received Data: ',firebase.get(URL))
            print('Recieved Data')
            print()
            break
        except:
```

```
            print('Reconnecting')
            print()

    print('X-axix',GyX)
    print('Y-axix',GyY)
    print('Z-axix',GyZ)


    print("Gps data: ",lat,long)
    print("Distance: ",parking_distance)



    sleep(0.5)
```

- **OTHER APPLICATIONS OF IOT**

Home Is Where the Smart Is:

EVM Machine-to-machine communication, and you understand you're not the most tech-savvy consumer, it's impossible that you've missed the abundance of home automation products filling the shelves and ads of every home improvement store. Suddenly an ordinary errand for light bulbs will leave you wondering if your lamp could send you amessage alerting you that the light bulb needs to be replaced. Furthermore, if your lamp is talking to you, could your refrigerator and sprinkler system be too? Experts say: Yes, the possibilities are endless. If that's the case, where do you begin?

Any day-to-day, repeatable process is automatable with smart home applications. The greater the control and flexibility of these processes, the more energy and cost savings the resident experiences, which are factors anyone who pays utilities strives to moderate. The smart home revolution is likely to be more of an evolution, with the incorporation of one or two home systems at a time, gradually automating our households through smart mobile devices.

However, with these elements of efficiency comes the question of ease of use. Will it bring you enjoyment or exasperation? With so many brands and models already available in an ever- growing market, how do you know which is best for you?

Lighting Control: Leaving the Dark Ages and Stepping Into the Light

Smart lighting allows you to control wall switches, blinds, and lamps, but how intuitive is a lighting control system? Itturns out, quite; its capabilities are extensive. You're able to schedule the times lights should turn on and off, decide which specific rooms should be illuminated at certain times, select the level of light which should be emitted, and choose how particular lights react through motion sensitivity, as seen with Belkin's WeMo Switch + Motion, which isboth affordable and easy to use with its plug-and-play simplicity.

HVAC Regulation: No Longer Burned by Your Heating Bill:

As fuel costs rise and the availability and sustainability of our resources becomes a greater concern, heating/cooling our homes efficiently is less of a budgetary bonus and more of a necessity. Over the past year, smart thermostats and automated home heating systems have become more readily available and easily incorporated into any home. Heating and cooling our homes consume an average of 50% of energy costs yearly, making daily HVAC regulation progressively rewarding. Maintaining a substantial lead among the nearly non-existent competition, the Nest LearningThermostat, learns your heating and cooling preferences over time, eliminating the need for programming and is accessible from your smartphone app. With automated HVAC you are able to reduce the heat when a room is unoccupied, and increase or decrease it at specific times based on your schedule and occupancy.

Smart Appliances: What's for Dinner?

Will smart kitchen appliances actually make you a better cook? Maybe. Smart refrigerators, such as LG's Smart ThinQ, allow you to scan grocery store receipts and keep an inventory of your items, and alerts you if an item is about to expire. More impressively, it suggests recipes based on your refrigerator's contents and lets you know when you need to replace items. Smart ovens sync with your smartphone and automatically preheat to the correct temperature based on a recipe selected from your database. While these appliance options seem a bit superficial and convenience based, there is a conservation factor as well. By automating your kitchen appliances and making them accessible from your smart device, you're able to sever the electricity supplied to unused appliances and reduce your energy consumption and costs. Considering the number of appliances the average household owns; this could save a substantial amount of money

Security Systems: Knock, Knock…

Who's there? The Internet of Things. While efficiency and conservation are certainly IoT benefits, its potential to have improved control over home security is a primary focus. Smart locks, like Kwikset's Kevo, a Bluetooth enabled electronic deadbolt, and various connected home security systems, such as iSmartAlarm, offer a variety of features including door and window sensors, motion detectors, video cameras and recording mechanisms. All of which are connected to a mobile device and accessible via the cloud, thus enabling you to access real-time information on the security status of your home. Naturally, there is a great deal of scrutiny regarding the level of trust in controlling your home's security system via a mobile device, but it begs earnest exploration when weighing the potential benefits and peace of mind it provides homeowners.

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 ESP-32 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 is needed 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.

REFERENCES

- Wikipedia.
- ESP-32 Data Sheet.
- www.electronicshub.com
- https://www.electrovigyan.com/arduino/sg90-servo-motor/