IOT BASED CONTROLLING DC MOTOR

ESW-PROJECT

Mallika Subramanian – 2018101041

Tanvi Karandikar – 2018101059

Jayati Narang – 2018101066

INDEX

Design Documentation

***Abstract*—The project is an IOT controlled motor. The motor is either controlled based on how close an obstacle/object is or can be controlled remotely. The micro-controller used is the ESP32.**

I. Introduction

A. Problem Statement and Scope

The proposed system allows remote controlling and monitoring the working of the motor. This concept can be scaled further to heavy industrial machines.

B. Purpose of the System

The internet is the most widely used, high speed and easily accessible communication medium in the modern day world. The objective of this project is to control a motor by varying its speed and observe and record the changes in certain parameters such as speed, proximity etc.

C. Overview of the System

The speed of the motor will be controlled via Pulse Width Modulation. An arduino code is flashed onto the board that manipulates the speed of the motor. The motor will be the central node interacting with the cloud using the OneM2M protocol as well as ThingSpeak thereby storing the readings and data on the cloud server. Data is pushed to the OM2M cloud server and ThingSpeak every minute. In order to remotely control the motor, the ESP32 board is made the server and a web page (UI) is rendered at the client side. Whenever the user presses one of the four buttons on the screen, a GET request is made thereby extracting the action to be performed and accordingly changing the behaviour of the motor.

II. System Requirements

The two main components of this project are the ESP32 micro-controller board -that acts as the central node- and the 150 RPM motor. In order to measure the distance of the objects in the vicinity of the motor proximity sensors are required.

To measure the speed of the motor, a photoelectric rotary encoder is used. A motor controller board (IC) is used to rotate the motor in either clockwise/anticlockwise direction. The entire setup is integrated using jumper wires and is soldered onto a PCB.

A. System Specification

- ESP32 Micro-controller board
- 150 RPM motor
- L239 chip
- Infrared beam sensor
- SR04 Range finder

B. Stakeholders

Scaling the project further the applications of this motor can be used in several fields.

- Industrial Companies for Machinery:
  It can be used in order to control the smooth functioning of machinery. Eg: A cutter equipment. This can be controlled by the proximity of objects thus preventing any injurious hazards and can further be controlled remotely in case of any defects in the equipment.
- Vehicles - autonomous cars:
  Another use case is in autonomous cars. This reduces the risks of accidents as the obstacles are detected by the proximity sensors and the speed of the motor (wheels in this case) is slowed down.

C. Design Entities

*1) **Micro-controller**:*

ESP32 is highly-integrated with in-built antenna switches, power amplifier, low-noise receive amplifier, filters, and power management modules. It adds priceless functionality and versatility to any applications with minimal Printed Circuit Board (PCB) requirements.

Programming an ESP32 device is as easy as programming an Arduino device. It can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. It can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces. Another huge advantage is that it not just supports the latest BLE Bluetooth 4.2, it also supports classic Bluetooth. This means it can speak to old and new Bluetooth phones/tables.

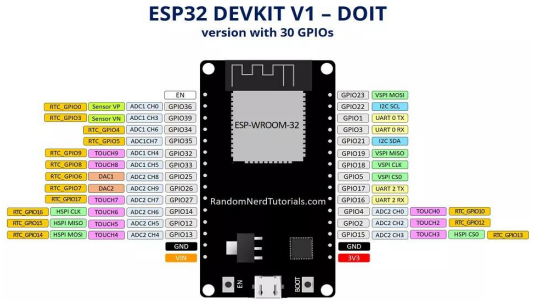

Fig. 1. ESP32

The board is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitry, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions. Furthermore, Espressif has added a dual-core CPU, larger RAM, more internal memory, and a bunch of hardware functions. The resulting system is powerful to answer the needs of modern IoT devices when we think of security, performance, energy efficiency and physical dimensions.

Specifications summary :

- Dual core processor.
- It has Wi-Fi and Bluetooth built-in.
- It runs 32 bit programs.
- The clock frequency can go up to 240MHz and it has a 512 kB RAM.
- This particular board has 30 or 36 pins, 15 in each row.

Fig. 2. DC Motor

- It also has a wide variety of peripherals available, like: capacitive touch, ADCs, DACs, UART, SPI, I2C and much more.
- It comes with built-in hall effect sensor and built-in temperature sensor.

*2) **150 RPM DC Motor**:*
A standard DC motor has been used for the demonstration of the project. Voltage: 4V to 12V

*3) **Motor speed sensor**:*

A motor encoder is a rotary encoder mounted to an electric motor that provides closed loop feedback signals by tracking the speed and/or position of a motor shaft. The Photoelectric Speed Sensor Encoder Coded Disc code wheel is a Slotted Opto isolator module, with an IR transmitter and a photo-diode mounted on it. Performs Non-Contact Object Sensing.

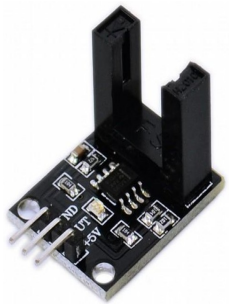

Fig. 3. Motor Speed Sensor

This is normally used as positional sensor switch (limit switch) or as Position Encoder sensors used to find the position of the wheel. The module consists of IR LED and Photo diode mounted facing each other enclosed in the plastic body

*4) **L293 Chip – Motor driving chip**:* L293D is a typical Motor driver or Motor Driver IC which allows DC motor to drive on either direction. It is a 16-pin IC which can control a set of two DC motors simultaneously in any direction. It means that you can control two DC motor with a single L293D IC. The L293D can drive small and quiet big motors as well. It works on the concept of H-bridge. H-bridge is a circuit which allows the voltage to be flown in either direction. Voltage needs to change its direction for being able to rotate the motor in clockwise or anticlockwise direction, Hence H-bridge IC are ideal for driving a DC motor.

Fig. 4. L293 Motor Driving Chip

*5) **SR04 Range finder-(Ultrasonic range finder)**:*
This HC-SR04-Ultrasonic Range Finder is a very popular sensor which is found in many applications where it requires to measure distance and detect objects. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object.It has very handy and compact construction. It offers excellent range, accuracy and stable readings in an easy-to-use package. Its operation is not affected by sunlight or black material

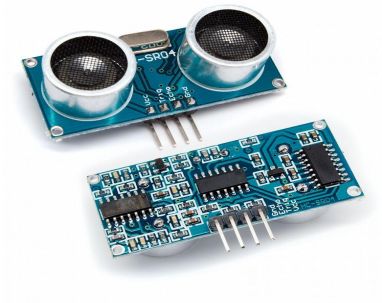

Fig. 5. SR04 Proximity Sensor

Specifications:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: ¡15°
- Operating Current: ¡15mA
- Operating Frequency: 40Hz

D. *Design Details (Entity Interaction)*

- The ESP32 (board) has a WiFi chip enabling it to communicate with the other nodes in the project.
- The board is connected to the motor controller board which in turn is connected to the motor.
- There are 2 proximity sensors attached to the motor. Each of these senses if an object is close to the motor and accordingly varies its speed by slowing it down.

- The photoelectric speed sensor is placed such that the wheel of the motor cuts through the IR radiation and enables the calculation of the speed of the motor in rpm.
- The Micro-controller board creates its own sever at a particular IP address in order to communicate with the user interface to GET and POST requests for remote controlling of the motor. The user here acts as the ”Client” and the ESP32 as the ”Server”.

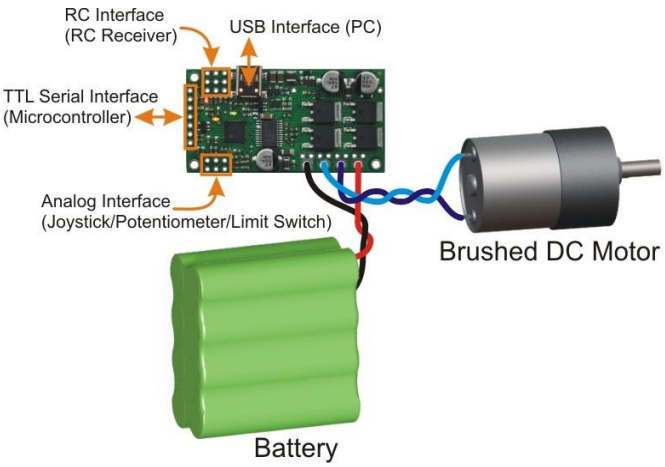

Fig. 6. Integrated components

E. *Operational Requirements*

*1) **System Needs**:*
The system requires a constant power source to keep running. In order to post the data that is collected to the ThingSpeak and OneM2M Servers, the ESP32 board requires a stable WiFi connection as well.

*2) **UI Design**:*
A user interface has been provided so as to enable the user to control the motor remotely and hence vary its speed as and when required.

The user can choose between the four modes of operation of the motor:

- STOP: To stop the running motor
- SLOW: To run the motor at a slower speed
- FAST: To run the motor at a fast speed
- GO : To run the motor as per the proximity of objects around it.

A dashboard with the graphs obtained for all the three parameters- distances, RPM- is also created for the convenience of the user. The dashboard supports real-time updation of the ThingSpeak graphs. In order to show the latest graphs from the OM2M server containers, first the python script to scrape the data must be run after which the new graphs will be plotted.

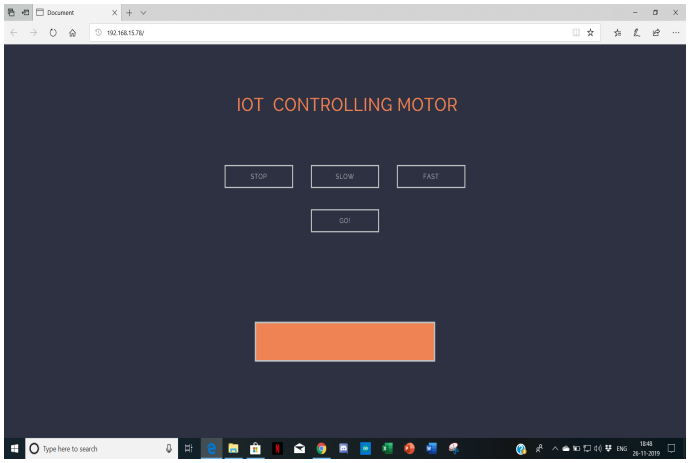

Fig. 7. User Interface Page

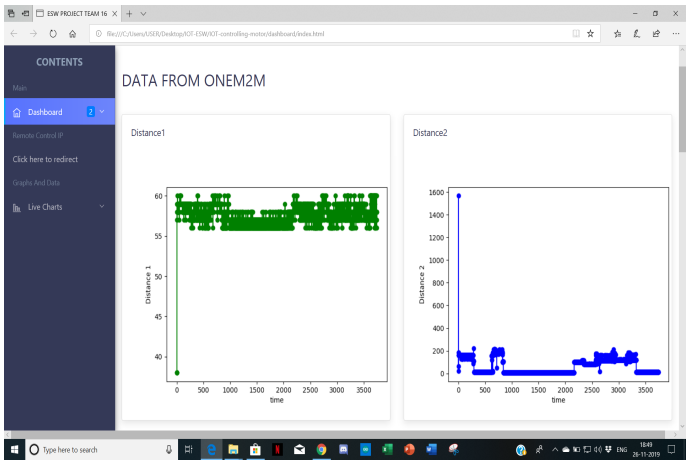

Fig. 8. Dashboard Snippet

*3) **Analytical System**:*
The readings of the two SR04 sensors and the Rotary encoder are pushed every minute to the OM2M server as well as the channel on ThingSpeak. The data that has thus been obtained is plotted on 3 separate graphs in case of the ThingSpeak channel. Further, using a python script the data from the OM2M containers is scraped and graphs for the same three parameters have been plotted as well. Data analytics performed for the data collected for the last one month include:

- Daily Traffic for a week
- Hourly Traffic for a single day

The graphs for the same are in Fig.9

F. *Communication*

The data collected from the sensors is pushed to the cloud (ThingSpeak and OM2M) via the WiFi network that the ESP32 board is connected to.

As for the client-server setup that has been established by the ESP32 server, it uses the HTTP communication protocol.

The flowchart in Fig 10 explains the same.

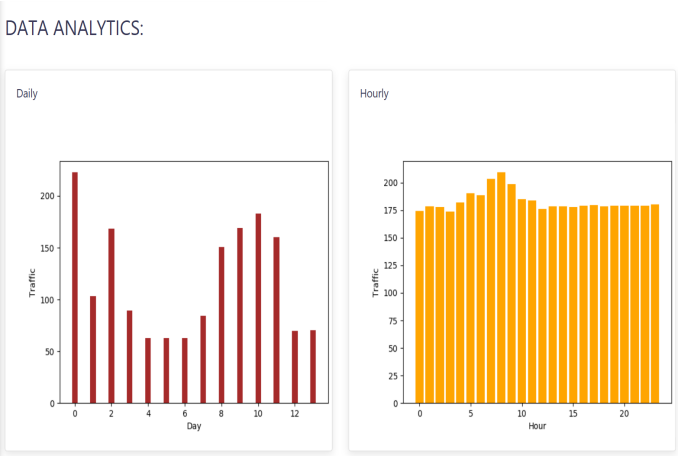

Fig. 9. Data Analytics

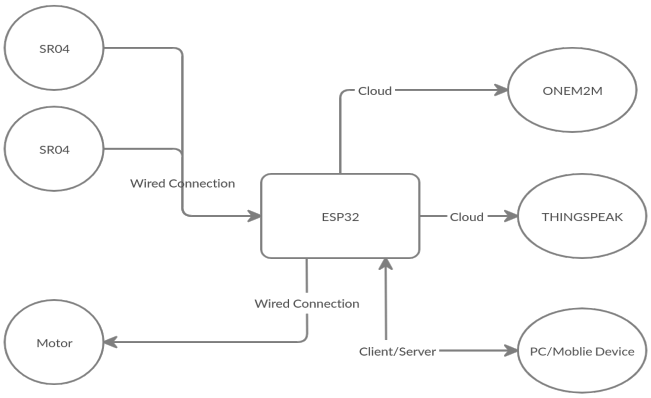

Fig. 10. Communication Flowchart

G. *Software Specifications*

- Arduino : 1.8.9
- Arduino Libraries
  - WiFi.h
  - HTTPClient.h
  - ArduinoJson.h
  - ThingSpeak.h
  - Arduino.h
- Arduino Board Manager: http://dl.espressif.com/dl/package_esp32_index.json
- Python : 3.7
- Eclipse OM2M : V1.3.0
- Java : 8.0

Technical Documentation

I. SENSORS' EXPLANATION

A. *SR04*

The HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and receiver. The sensor works with the simple formula:

$$Distance = Speed \times Time \tag{1}$$

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below. Now, to calculate the distance using the above

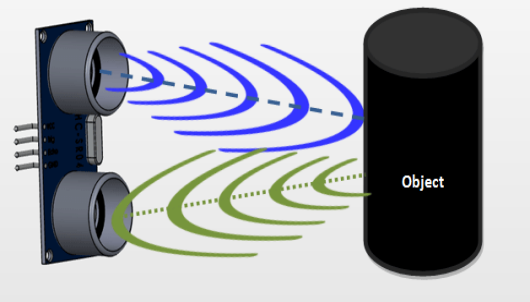

Fig. 1. SR04 sensor

formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now, simply calculate the distance using a microcontroller or microprocessor.

B. *Optical Speed Sensor*

The sensors send speed data in the form of electrical pulses. Shaft encoders offer a high resolution of typically 1-5000 pulses per revolution (PPR) and clearly defined, symmetrical pulses. In principle, RPM sensors convert mechanical motion into electric pulses with or without direct contact when positioned near a turning rotor, gear, shaft or other regularly moving device. The resultant output signals are then fed to a digital counter, totaliser, tachometer, or other monitoring and control device. In this sensor, rotation is transmitted to the measuring instrument via an infrared (IR) light beam/laser beam coming from the instrument, which is then reflected by a reflective tape on the object. As explained, traditional tachometers require physical contact between the instruments and the objects being monitored. Laser tachometer is a powerful choice where this type of direct-contact type measurement is not workable for technical or safety reasons. Laser tachometers work by pulsing a laser beam against the rotating element. The rotating element

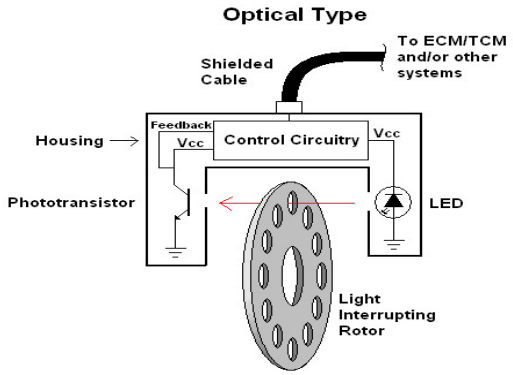

Fig. 2. Infrared speed sensor

will have one reflective spot, and the tachometer measures the rate at which the light beam is reflected back.

II. OM2M

OM2M is a global standardization for M2M communication and IoT. It provides a software framework by creating a horizontal layer across domains. OM2M enables reusability through interoperability.It is located in the transport layer in the networking stack.

With Respect to this Project:

The data collected by the sensors is to be stored on a server. The chosen protocol here is OM2M. Every minute the sensor values are pushed to this server.
First a root node (Common Service Entity) for the resource tree is created. Then a container for the ESP32 is created within which we have a container for the sensor values that are pushed every minute.
To be able to access the server and push new data we require the cse_ip and the uri of the container. Once this is done, to send a fresh set of data we have to create a new container.
The appropriate headers are added to the packet of data that is to be sent (X-M2M-Origin, content type, content length) after which a post request is made to the server.The newly updated container will now appear in the resource tree.

Attribute	Value
rn	cin_2138582
ty	4
ri	/in-cse/cin-2138582
pi	/in-cse/cnt-395057765
ct	20191029T085202
lt	20191029T085202
st	0
cnf	text/plain:0
cs	9
con	(178,8),0

Fig. 3. A OM2M container

III. THINGSPEAK

Just like the OM2M, data is simultaneously pushed to Thingspeak as well. To do so, a channel is created on thingspeak with a unique write API key and a channel number. Every one minute the thingspeak fields are updated and the new values are written into that particular channel. Thingspeak goes on to then produce graphs for all the data points that are obtained for each field.

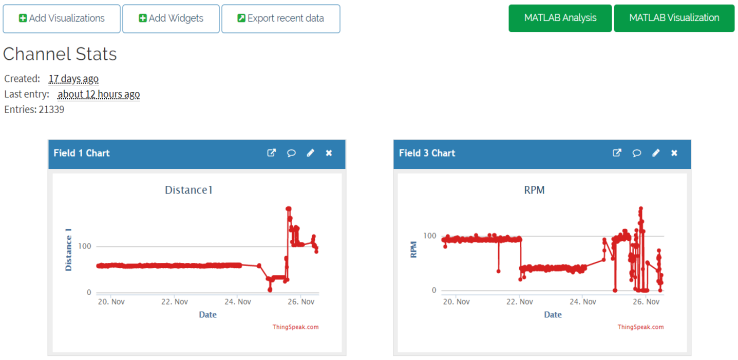

Fig. 4. An Eg ThingSpeak view

IV. REMOTE CONTROLLING

To enable a mobile device/PC to be able to remotely control the motor, a client-server relationship is to be established between the two. For this we create a server of the ESP32 board itself. The client will then be hosted on a specific IP address. A web page is rendered at that IP address. Through this a GET request can be made to the server by clicking one of the button options thereby asking for a change in the behaviour of the motor. The server side (ESP32 board) then decodes this request and changes the speed of the motor accordingly.

User Documentation

I. Introduction

The proposed system allows remote controlling and monitoring the working of the motor. This concept can be scaled further to heavy industrial machines.

II. Product Operational Requirements

A. *Dashboard*

To view the dashboard, the index.html static page should be rendered.

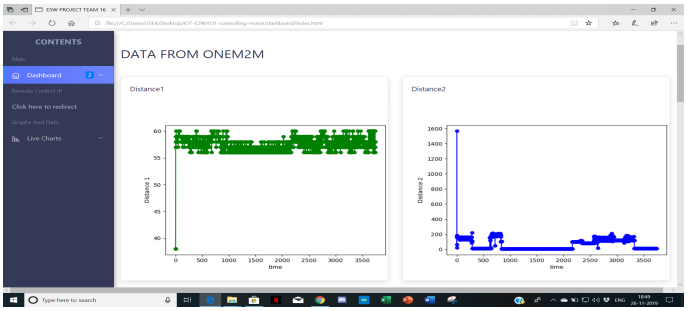

Fig. 1. Snapshot 1

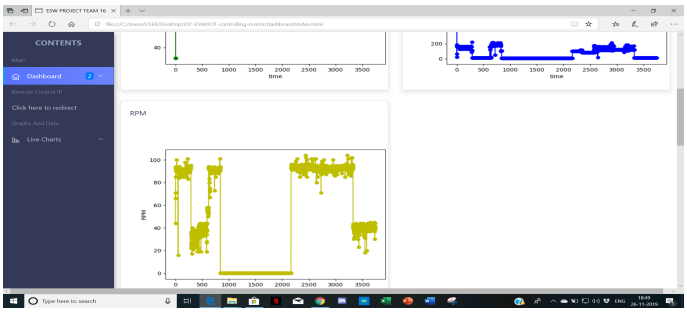

Fig. 2. Snapshot 2

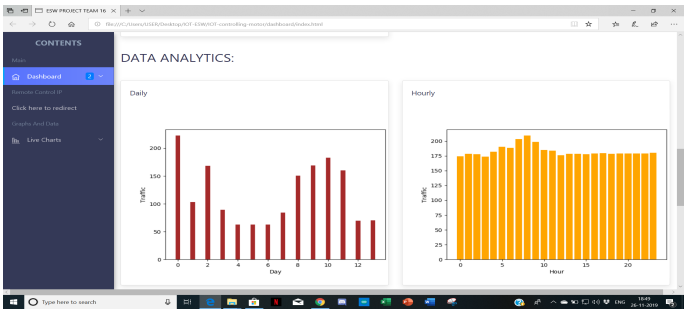

Fig. 3. Snapshot 3

B. *Remote Controlling*

Connect to the same WiFi as the ESP32 board which will be provided to you.

The IP address to access the control web-page is http://192.168.15.78/

Controlling the speed of the motor can be done in the following ways:

- STOP: To stop the running motor

- SLOW: To run the motor at a slower speed

- FAST: To run the motor at a fast speed

- GO : To run the motor as per the proximity of objects around it.

In any mode apart from GO, the speed of the motor won't vary based on the data of the sensors. The Page will initially open as shown in the figure. The orange box will show the current status of the motor.

Fig. 4. Snapshot 3

C. *Web Scraping*

The Dashboard shows real-time graphs from Thingspeak. However in order to view the latest graphs that will be obtained from the OM2M data, first the start.py python script is to be run. Following this run the update.py python script which will then update the graphs.