Ahmedabad University

ECE504: Internet of Things

Final Project Report

Section Number: 1

GROUP Name : **ERROR_404** | GROUP No.: **6**

Submitted to faculty: Anurag Lakhlani

Student Details

Roll No.	Name of the Student	Name of the Program
AU1940118	Raj Gariwala	B.Tech CSE
AU1940119	Nihar Patel	B.Tech CSE
AU1940151	Purvam Sheth	B.Tech CSE
AU1940171	Mohit Prajapati	B.Tech CSE

2022-2023 (Monsoon Semester)

Project Report -1

Introduction:

This project contains many IoT-based household appliances, which could be helpful to the homemaker in making a smart house. It is often observed in this fast life, where almost every house member belongs to the working class, so to improve their time efficiency while at home, the features offered by this project play an essential role in helping them. In addition, it is being experienced that people sometimes unintentionally forget to take care of some sensitive products, which surprisingly are handled in this project. The focus of this project is to make home smart and help by making day to day processes monitored and alerting individuals about some unexpected process. Our project can be helpful to ones who are living single at home and also to the homemakers as it helps to keep dreadful scenarios away, with the help of the alerts passed before hand in their mobile phones via Email or SMS. Last but not least this project helps households to reduce their carbon footprints.

Market Survey:

Product-1: Tata Power

Here, we have the first company who are selling smart solutions for home appliances. This company is currently performing very well because they are providing a one-stop solution to the customer by providing one app/website and also, they are focused on energy saving. Here we would like to add that we just should not focus on automation on electronic appliances and one would like to focus on how one should make home smart in a real micro level sense. For instance, You might have heard about Smart Refrigerators which can automatically order the food items which are running low in the refrigerator, so inspiring that we are here building a Raspberry Pi Smart Container using a set of sensors. This Smart Container can tell you about its status like whether it is full or empty, by sending a mail to your Email ID. Moreover, similar concepts we would like to apply on water level in water tank.

Product-2: Maximum Innovation

Here, we have a company, Maxxio, which provides smart home appliances. In this product, the company provides individual displays for all of the appliances which results in a higher cost of the product. Instead of using display, they should focus on integrating all the features and make an app/website that can work as a one-stop solution and also it is more feasible to customers in terms of pricing and availability. So, someone who is not at home can also run appliances using the app/website on their device.

Product-3: UGVCL Online

As we are living in the technological world, there are several steps taken by the government to make India, digital India, and owing to that there are several sectors moving digital and one of the examples is UGVCL (Uttar Gujarat Vij Corporation Limited) - government body, electricity provider in Uttar Gujarat, which displays the bill and the energy consumed in the each home in their application and update its information every month. However, daily statistics, billing and consumption details are not available in the application. Hence in this project we are likely to introduce a IoT based electric energy meter in which we can monitor our electricity uses from anywhere in the world and get an SMS/E-mail when your energy consumption reaches a threshold value and control certain appliances which might be consuming more energy with just a click from a dashboard. Hence here we are trying to take one step toward sustainability with the help of technology.

Product-4: Schneider Electric

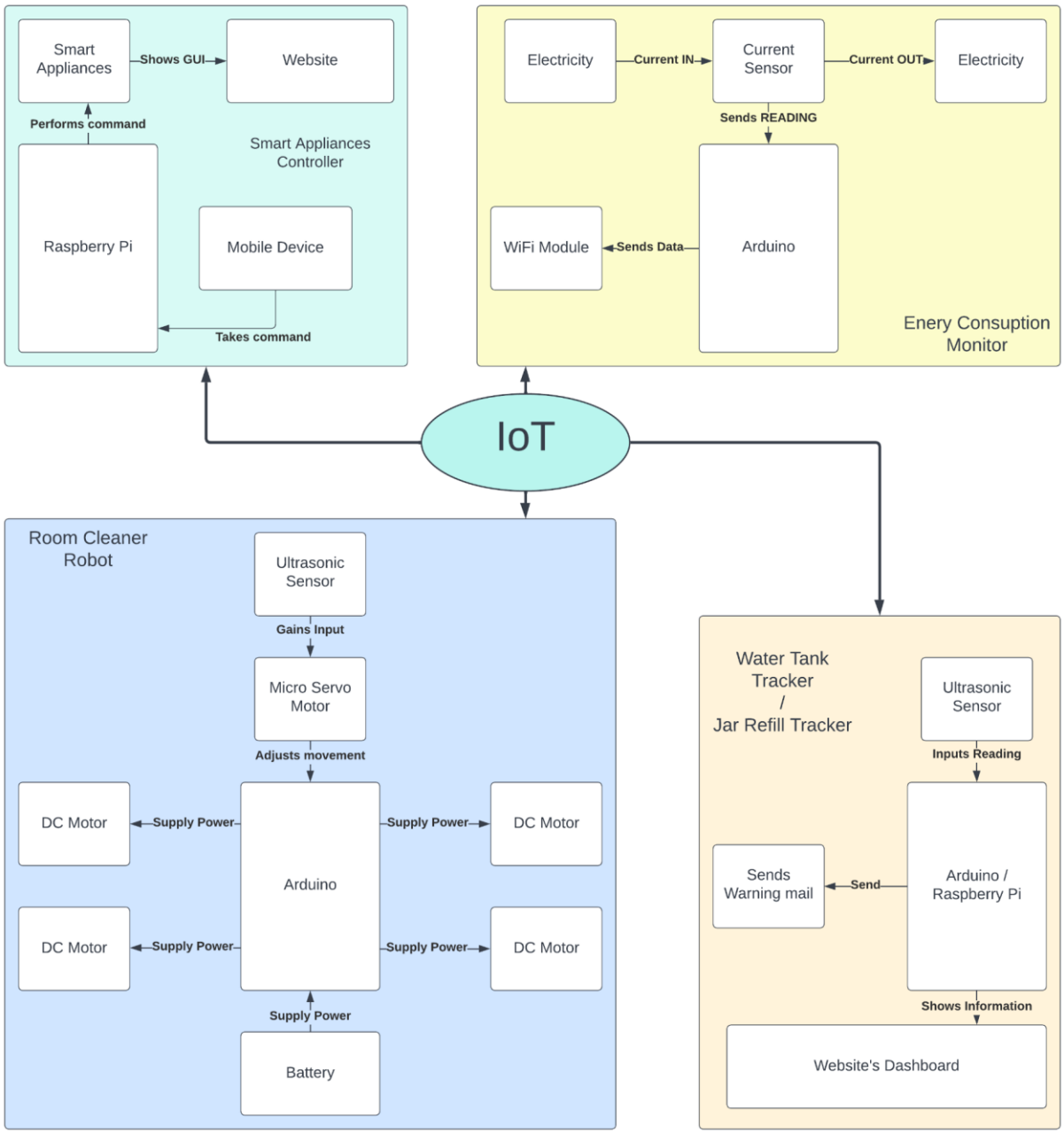
Here, we have the third company which provides smart home appliances, and that too integrated into one application. However, here they are lacking different features which they could add to it. We will not only provide these features but also add smart devices which will focus on energy consumption. Users will be able to track the energy consumption done by each device which helps them to monitor the uses of the devices. The more we save energy today, the better it is for the next generation.

Product-5: XIAOMI

There are already full-fledged products available in the department of automated cleaning robots using the combined power of IoT and AI, with the latest and advanced hardware technologies. Most of the manufacturers have made these cleaning robots a premium product for their sales.

Currently a brand which we all know for its low and budget friendly prices for their products, from the rest of the market –XIAOMI(MI). MI, which is very well known for its budget friendly products, starts its prices from Rs. 24,999. Another company which is well known for its consistency and its reliability in the long run –SAMSUNG. Samsung starts its product line for Rs. 49,000. This high price is the result of highly expensive hardware used in it like LIDER, motion sensor, charging station etc.

Block diagram and Explanation:



Explanation:

· IoT-Based Electricity Energy Meter: This product helps in monitoring electricity use anytime and from anywhere in the world, which makes this feature useful. The feature of sending an email is also integrated and we would demonstrate that with just one click how we can control appliances and save energy.

· IoT-Based Smart Container with Email Alert and Web Monitoring: This Smart Container can tell you about its full or empty status by sending a mail to your Email ID. It can also monitor the weight of the container in real-time using a web browser, which makes it an IoT project where one can monitor the container from anywhere using the internet.

· IoT-Based Web Controlled Home Automation: This home automation system allows you to operate your house's equipment from anywhere in the world, such as by operating any AC appliances by clicking buttons on a webpage.

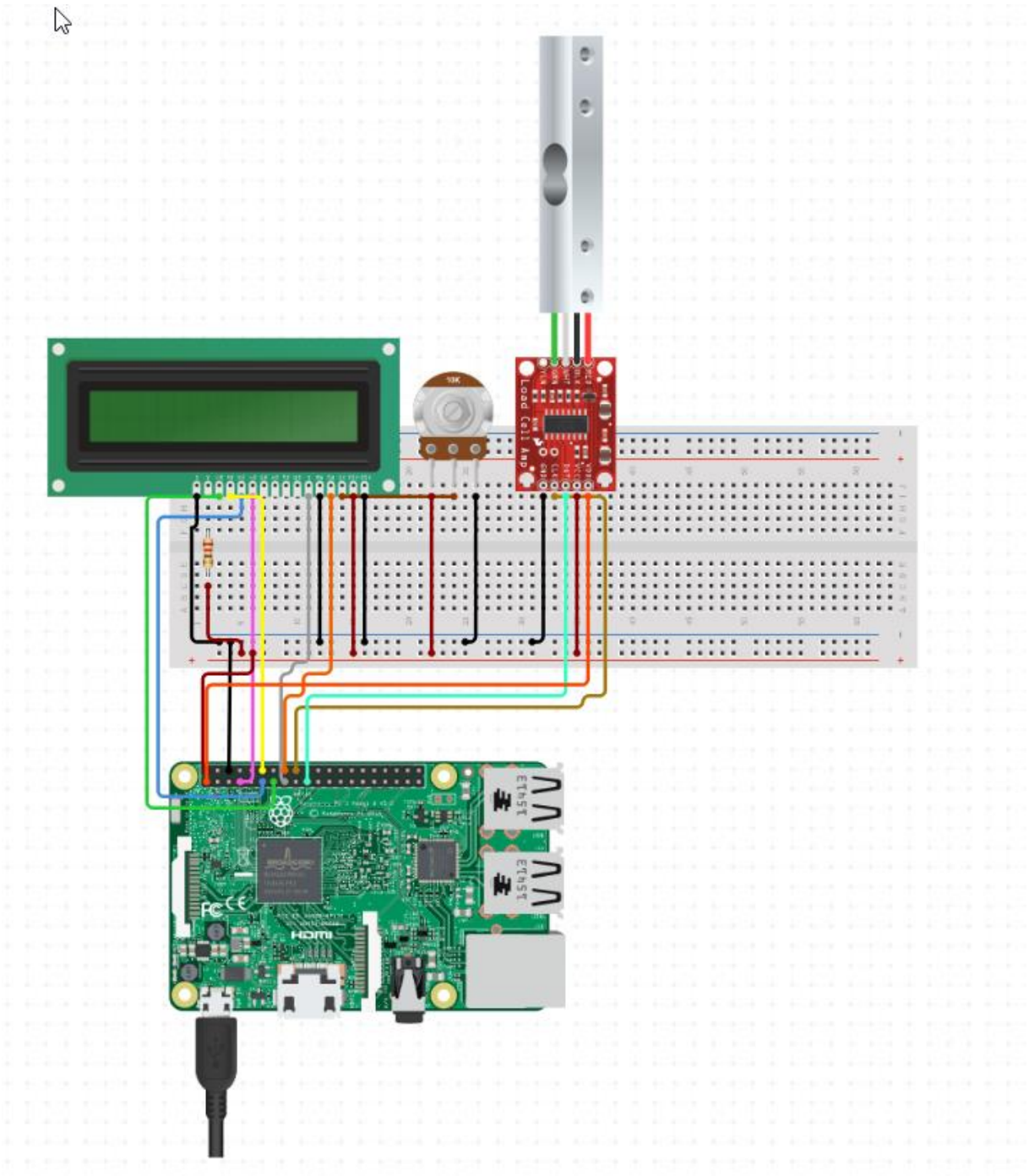
· Smart Water Tank with online monitoring dashboard: This device will help the user to monitor the level of his/her water tank. Any person can easily track the water level of the tank. If it decreases beyond a certain level, the system will send a notification with an auto-generated message to the person to refill the tank.

· Smart House Cleaning Robot: This robot will make the cleaning processes (like sweeping and mopping) automated with the help of a single touch command from your device. This can be possible with the help of an Ultrasonic sensor.

Project Report -2

Chapter: 4 – Circuit Diagram

IoT Raspberry Pi Smart Container:

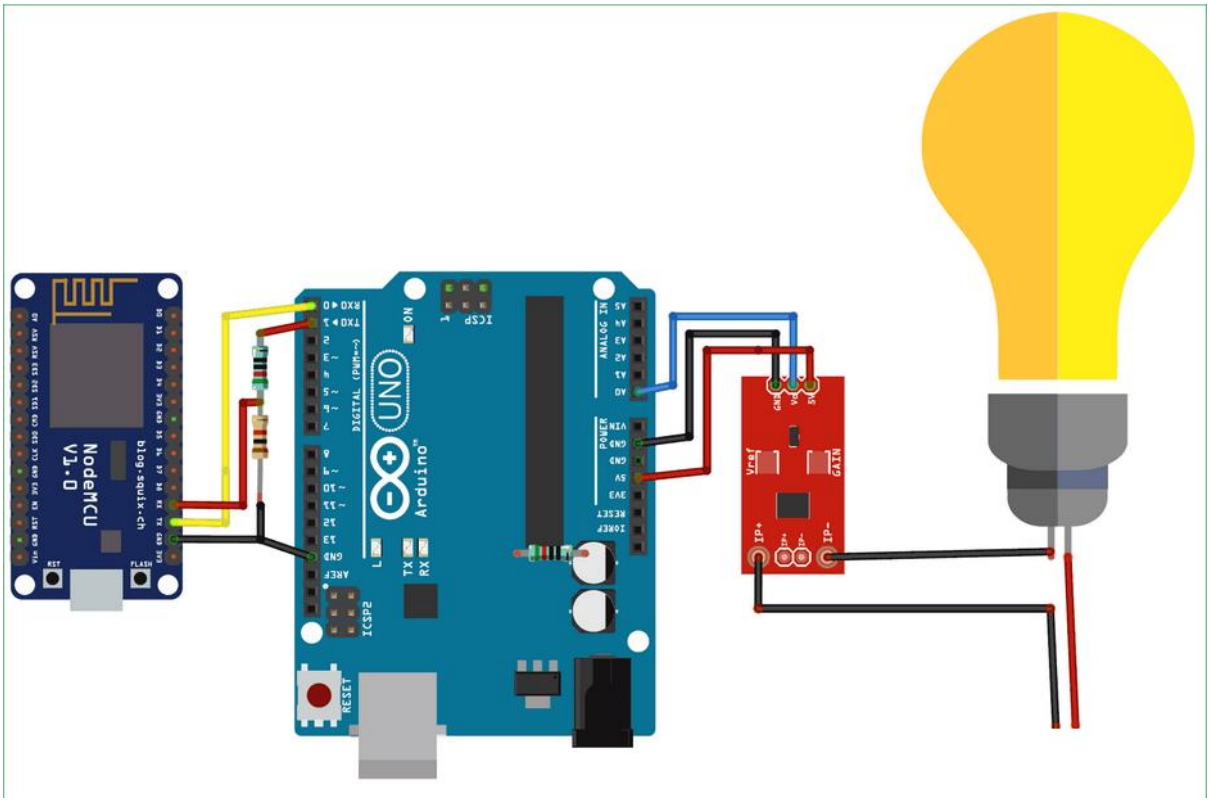


This project will help you to check the status of your Jar (How much Jar is filled) using the internet since it has the ability to send you an email when it is full or empty and has the ability to measure the weight of the Jar in real-time and show results on a web service. Here, the weight of the container will update on the web browser every five seconds; this interval may easily be modified in the HTML code file. The "Container is Full" email notification threshold weight value has been set at 300 grams, although this restriction is movable.

It's simple to use this Smart Container. In this project, the entire process is controlled by a Raspberry Pi 3. The HX711 Load Amplifier Module receives an electrical analog voltage from the load cell, which senses the container's weight. The HX711 is a 24-bit ADC that amplifies and converts the load cell output. The Raspberry Pi is then given this value that has been magnified. The HX711 output is now calculated by Raspberry Pi and converted to a weight value. Then, using Python code on a Raspberry Pi, this weight number is compared to the already set threshold value (e.g., 500 grams). If the weight of the Jar exceeds 500gm, System sends an email stating, "Smart Container Alert… Container Full." And if the weight of the Jar remains under 500gm then System sends an email stating *"Smart Container Alert.... Container is Empty".* The weight of the Jar will also be shown to the user using a web service, and a 16×2 LCD display.

Why we have selected this sensor:
Learn how this sensor module functions first. A breakout board called the HX711 amplifier makes it simple to read load cells and measure weight. You served as the microcontroller on one side and the load cell wires on the other. Using a two-wire interface, the HX711 connects with the microcontroller (Clock and Data). This is the reason we decided to work with this sensor in the first place.

IoT-based electricity energy consumption tracker:

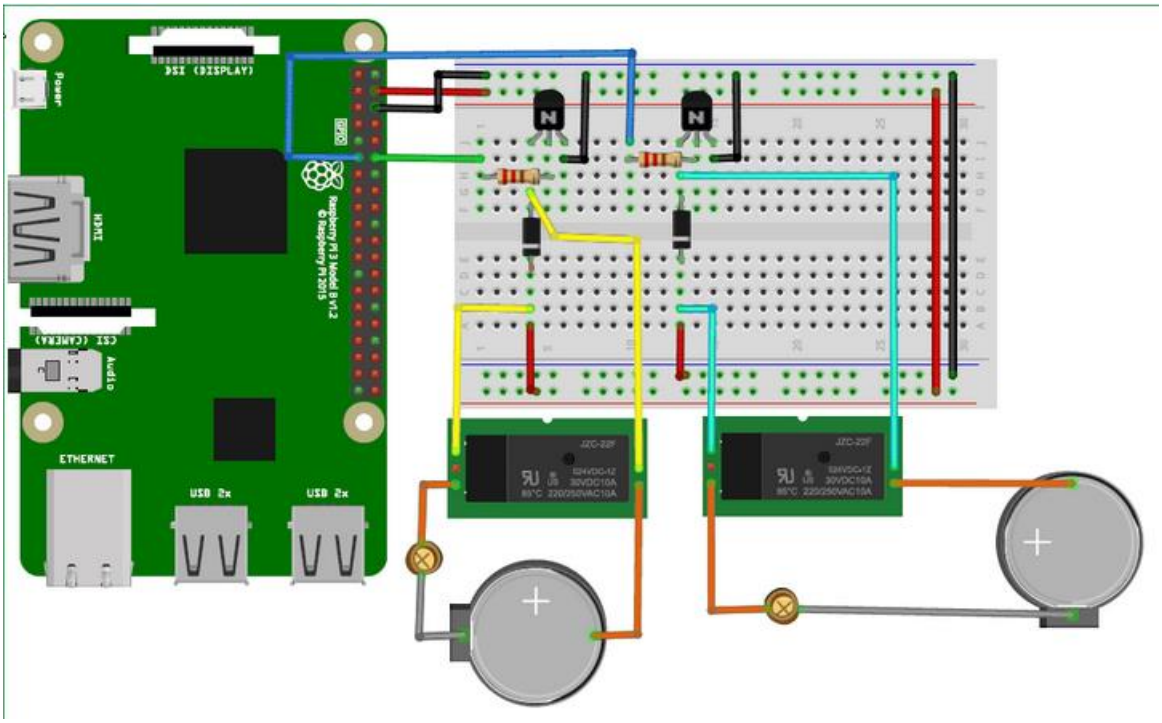


NodeMCU has one analog pin (ESP12), which we might utilize, however, the ESP series can handle up to 3.3 volts on its pins. We are not utilizing a standalone NodeMCU because the current sensor we are using, which has a maximum output of 5 volts, has the potential to destroy our Wi-Fi module. In order to change the current sensor's output from 5V to 3.3V because, as we explained previously, the current sensor's output current is 0Amp at 2.5Volts. As a result, the Arduino will use an analog pin to read the value of the current sensor and then transfer it through serial communication to the ESP12 Wi-Fi module. Use voltage divider circuit at receiver pin of NodeMCU so that receiver pin can get up to 3.3 Voltage level. Now, this input is taken by the Arduino circuit and it will generate output using a computer program and will show this output on the dashboard of the user's system.

Why we selected the ACS712 sensor:

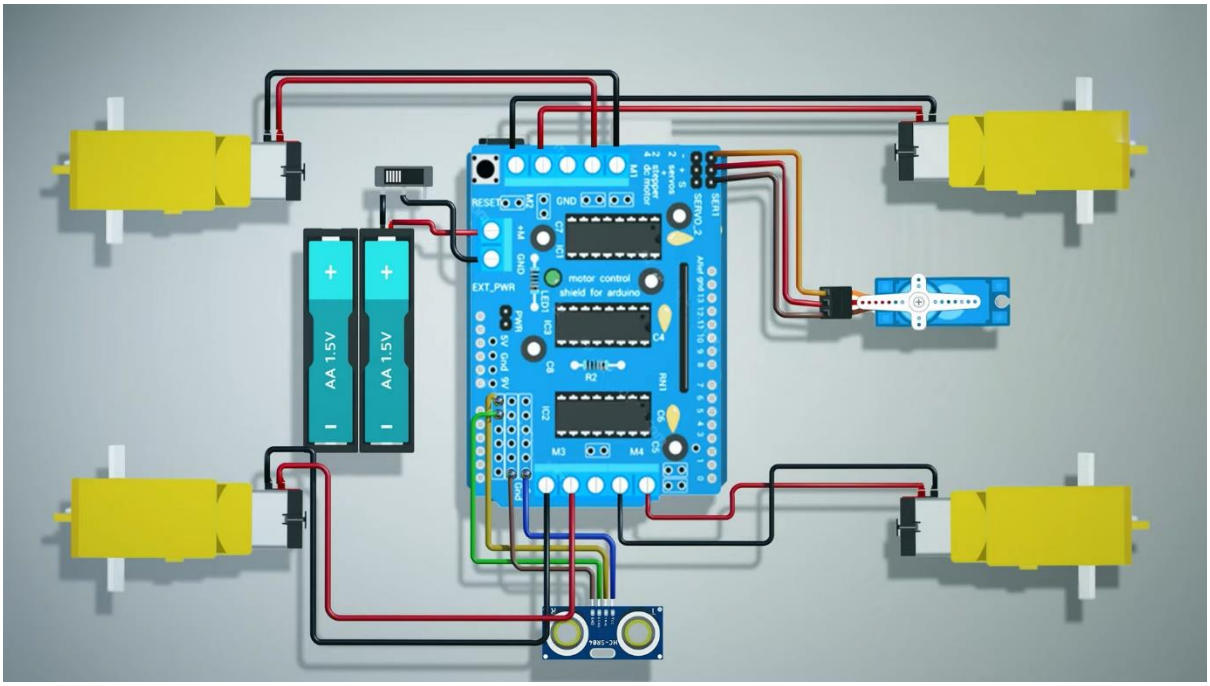
Since the ACS712 Current sensor is a crucial part of the project, it is crucial that we comprehend its operation before we begin construction. Due to the noise that comes along with it, poor isolation issues, and other factors, measuring current, is always a difficult process. However, things have gotten a lot simpler with the help of this Allegro-designed ACS712 module. The ACS712 Current Sensor's ability to detect both AC and DC current as well as its provision of isolation between the Load (AC/DC load) and the Measuring Unit are its main benefits (Microcontroller part).

IoT-based Home Automation:



Here this project helps us to turn on and off different switches from one website. Here we will use the WeblOPi service which will act as an intermediate as we take inputs from the website and it will give instructions to the pi. Using this input Pi will decide whether to give the current to the specific circuit or not. So, suppose the user gives input from a web application to turn on the light present in the bedroom. Pi recognizes this input and makes changes to the circuit and turns on that specific light.

Smart Vacuum Cleaner Robot:



This smart vacuum robot will be cleaning the room with the use of its vacuum and store the trash and dirt in the container attached to it. Here in this case this circuit will take the input from the ultrasonic sensor and will send it to the Arduino circuit and based on this input, the code will give instructions to change the direction. If we talk about the cleaning part then it really needs no input as we have to only make sure that the motor is running or not.

Why we have selected the ultrasonic sensor:
Ultrasonic sensors are used as proximity sensors. This sensor is used in different areas like automobile self-parking technology and anti-collision safety system, robotic obstacle detection system, etc. One of the main reasons to select the ultrasonic sensor is that it is highly accurate and can be used to detect very small alterations in position. Due to the high accuracy, and cheap prices, we are using this sensor and also it will give us the desired output we wanted.

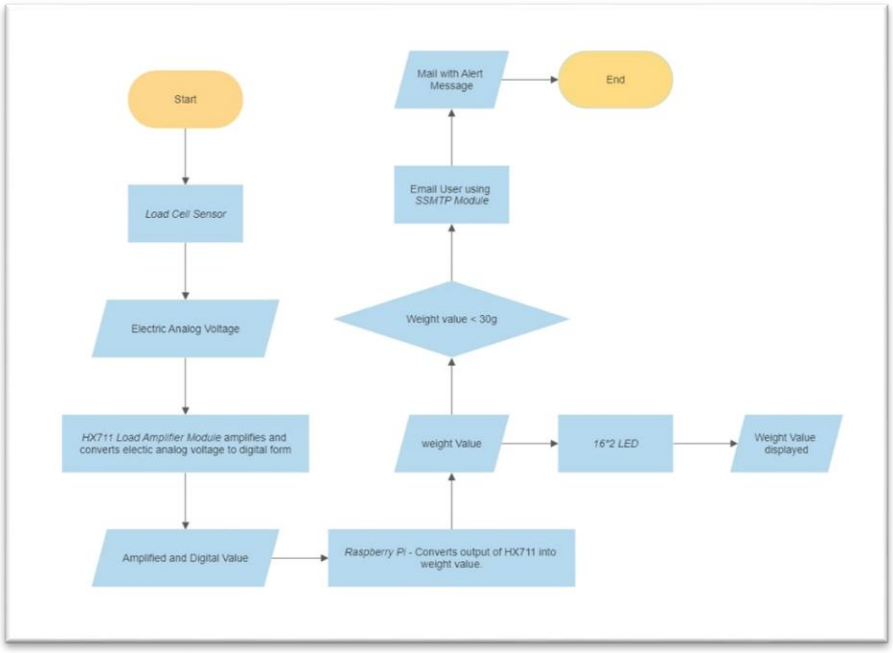
Chapter: 5 - Table of Comparison

Arduino UNO	Raspberry Pi	BeagleBone	Intel's Galileo	Intel's Edison
A microcontroller board that is based on the ATmega328P, Arduino UNO has 14 digital I/O pins out of which 6 can be used as PWM outputs, 6 are analog inputs. Arduino UNO comes as a 16 MHz ceramic resonator, with a USB connection, a power jack, an ICSP header, and a reset button.	Raspberry Pi is an inexpensive computer that can run Linux, as well as it provides a set of GPIO (general-purpose input/output) pins, that allow the user to control electronic components used to measure physical computing as well as to explore the Internet of Things (IoT).	A barebone development board, BeagleBone comes with a Sitara ARM Cortex-A8 processor that runs at 720MHz, with 256 MB of RAM, comes with two 46-pin expansion connectors, with an on-chip Ethernet, a micros slot, along with a USB host port and a multipurpose device port, that includeslow-level serial control and JTAG hardware debug connections.	A Microcontroller board based on Intel Quark SoC X1000 Application processor, Intel Galileo, comes with a 32-bit Intel Pentium class system within the chip. This board comes with many computing industry standard I/O interfaces, that includes ACPI Express, PCI Express, 10/100 Mbit Ethernet micro SD or SDHB, USB 2.0 device and EHCI/OHCI USB host ports, high-speed UART, RS-232 serial port, along with a programmable 8MB NOR flash and a JTAG port.	A single-board computer platform, Intel Edison helps the user build things similar to an Arduino. Platforms like Arduino and Eclipse can also work with Edison. Edison uses an Intel Atom processor along with a dual-core CPU at 500 MHz and a microcontroller at 100 MHz. This board also supports 1GB of memory, 4GB of storage and also has Wi-Fi and Bluetooth 4.0 capabilities. It also comes with 40 GPIOs used for plugging sensors and motors.
Pros ✓ Ready to Use ✓ Examples of codes ✓ Effortless functions	**Pros** ✓ Vast Peripheral Support ✓ Multiple Sensors	**Pros** ✓ GPIO real-time units ✓ Onboard storage	**Pros** ✓ High CPU frequency than Arduino	**Pros** ✓ In-built connectivity option for Bluetooth

✓ Large Community	✓ Supports All Types of Codes ✓ Faster Processor ✓ 5. Can be used as a portable computer	✓ Analog inputs ✓ 4. Huge community availability		and Wi-Fi devices.
Cons 1. Structure 2. Cost 3. Easy to use (easily available things disables the ability of a person to understand the basics)	**Cons** 1. Missing eMMC Internal Storage 2. Graphics Processor missing 3. Impractical as a desktop computer 4. Overheating 5. Not able to run Windows Operating system	**Cons** 1. Small number of USB ports 2. Video encoding is lacking	**Cons** 1. Low CPU frequency than Raspberry Pi	**Cons** 1. Cost 2. Complicated and difficult use 3. No software control over motor currents
		Comparison with Raspberry Pi: 1. BeagleBone provides easy setup than Raspberry Pi: Beagle has a mini cable that helps in supplying power to the system, while Raspberry Pi does not have a micro-USB cable, as well as it does not have any pre-installed operating	**Comparison with Raspberry Pi** 1. Slow performance: When compared with Raspberry Pi, Galileo performs slower than Raspberry Pi and takes almost 2 times more time than Raspberry Pi, because the CPU frequency of the former is	**Comparison with Arduino** 1. It has two extension boards that allow Edison to be compatible with any other shield of Arduino. 2. It has better storage capacity than Arduino, which means more large and important files can be stored easily. 3. It has better power

		system. Also, the Beagle can boot from onboard storage, while Raspberry Pi requires an SD card to boot the system. 2. Number of connection points: The Raspberry Pi consists of 26 connection points for various interfaces whereas Beagle provides 92 connection points. 3. High-Speed processor: Beagle comes with a 1 GHz processor, while Raspberry provides 700MHz processor. 4. Different Architecture: The older version of Raspberry Pi has ARMv6 architecture, while Beagle comes with ARMv7 architecture.	less than the latter.	consumption than Arduino.

Chapter: 6 – Flowchart of Process



We used a Raspberry Pi 3 to manage the entire procedure. The HX711 Load Amplifier Module receives an electrical analog voltage from the load cell, which senses the container's weight. The HX711 is a 24-bit ADC that amplifies and digitizes the load cell output. The Raspberry Pi is then given this value that has been magnified. The HX711 output is now calculated by Raspberry Pi and converted to a weight value.

Then, using a Raspberry Pi and Python code, the value of the weight gets compared with the set value, the set value is 30g. If the weight remains below 30 grams, the Raspberry Pi sends an email indicating “Alert ... Container is Empty!”.

Chapter: 7 – Sensor

Load Cell Sensor

Operating Principle of Load Cell:

The load cell sensor works on the piezo-resistivity principle. When the sensor is loaded with weight, its resistance value changes, which leads to a change in output voltage when applied by an input voltage.

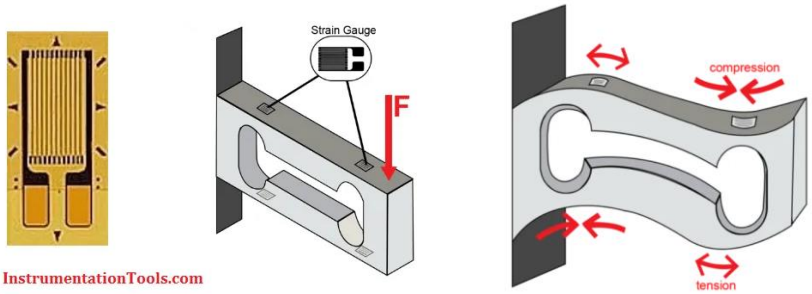


Figure: Operating principle of Load Cell Sensor

Reference 1: https://instrumentationtools.com/wp-content/uploads/2016/05/instrumentationtools.com_resistive-load-cell-working.png?ezimgfmt=ng%3Awebp%2Fngcb1%2Frs%3Adevice%2Frscb1-1

Reference 2: https://instrumentationtools.com/wp-content/uploads/2016/05/instrumentationtools.com_load-cell-indeflection-420x420.png?ezimgfmt=ng:webp/ngcb1

Physical Dimension:

Single Point load cells; are used in small to medium platform scales with platform sizes of 200x200mm up to 1200x1200 mm.

Others Specification:

SPECIFICATIONS

ACCURACY – (MAX ERROR)		
Static Error Band – %FS		±0.04
Nonlinearity – %FS		±0.04
Hysteresis – %FS		±0.03
Nonrepeatability – %RO		±0.01
Creep, in 20 min – %		±0.025
Side Load Sensitivity – %		±0.25
Eccentric Load Sensitivity – % / in		±0.25
TEMPERATURE		
Compensated Range	°F	+15 to +115
	°C	-10 to +45
Operating Range	°F	-65 to +200
	°C	-55 to +90
Effect on Zero - %RO/degree	°F	±0.0008
	°C	±0.0015
Effect on Span - %/degree	°F	±0.0008
	°C	±0.0015

Pin Diagram:

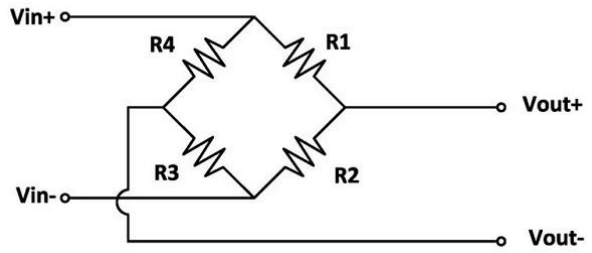
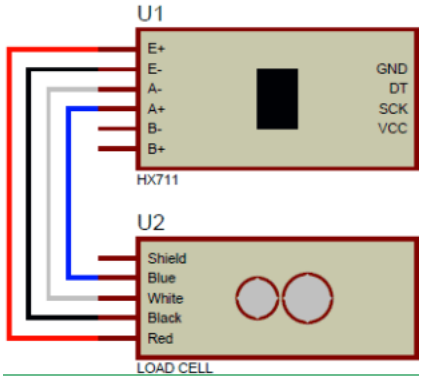


Figure: Pin Diagram of Load Cell

Image Reference: https://www.utilcell.com/upload/img1.jpg

Type of Interface: To make an interface with Raspberry Pi, the load cell needs to connect with an amplifier, which in this case is the HX711 Load Cell Amplifier Module. For this, the interfacing method used is I/O interfaces for sensors.

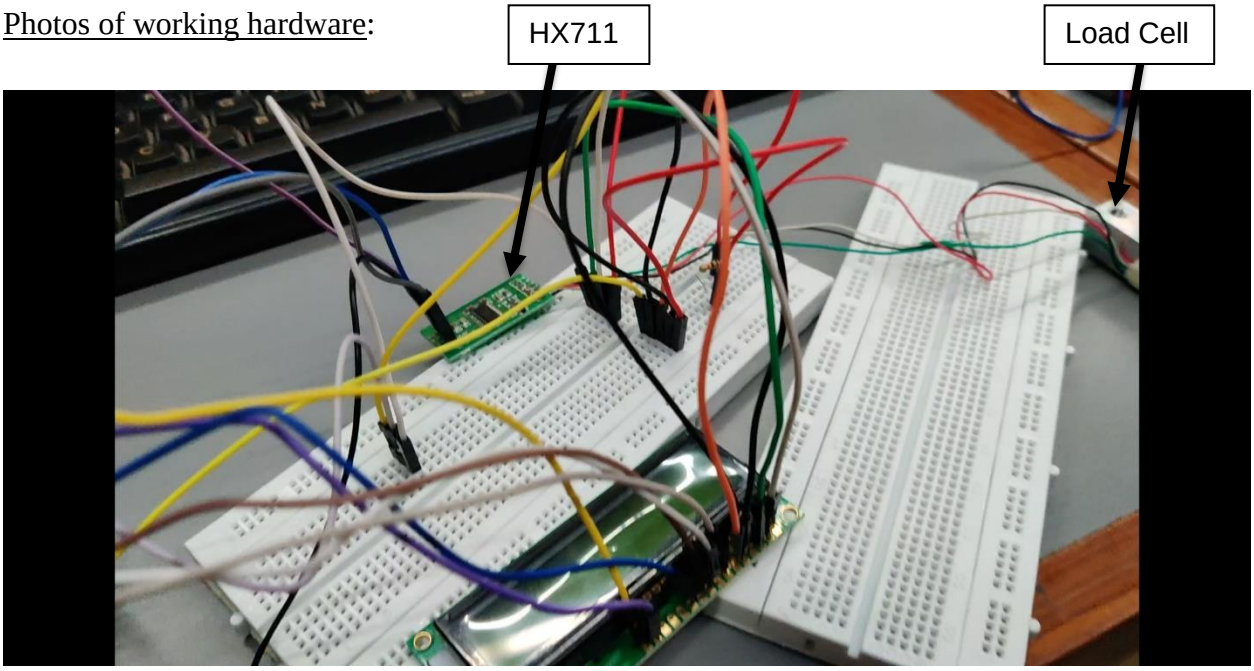


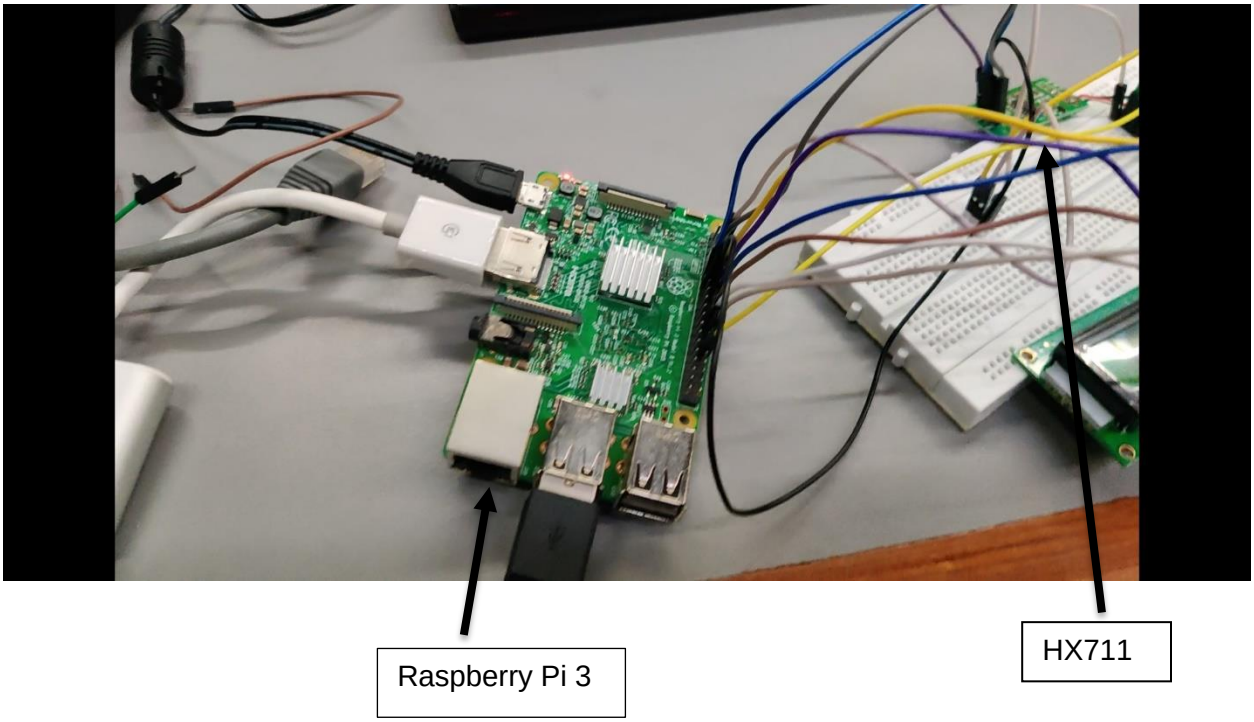
*Next: interfacing is mentioned in ch-8 i.e., for the actuator used (HX711 amplifier) **

Code to communicate with sensors:

```
def readCount():

  i=0

  Count=0

  gpio.setup(DT, gpio.OUT)

  gpio.output(DT,1)

  gpio.output(SCK,0)

  gpio.setup(DT, gpio.IN)


  while gpio.input(DT) == 1:

    i=0

  for i in range(24):

      gpio.output(SCK,1)

      Count=Count<<1

      gpio.output(SCK,0)


      if gpio.input(DT) == 0:

        Count=Count+1

        #print Count


  gpio.output(SCK,1)

  Count=Count^0x800000

  #time.sleep(0.001)

  gpio.output(SCK,0)

  return Count

```

Photos of working hardware:





Chapter: 8 – Actuator and Displays

HX711 Load Cell Amplifier Module:

Operating Principle of HX711 Load Cell Amplifier Module:

HX711 is an electronic scale module, which works on a principle to convert the measured changes in resistance value through the conversion circuit into electrical output.

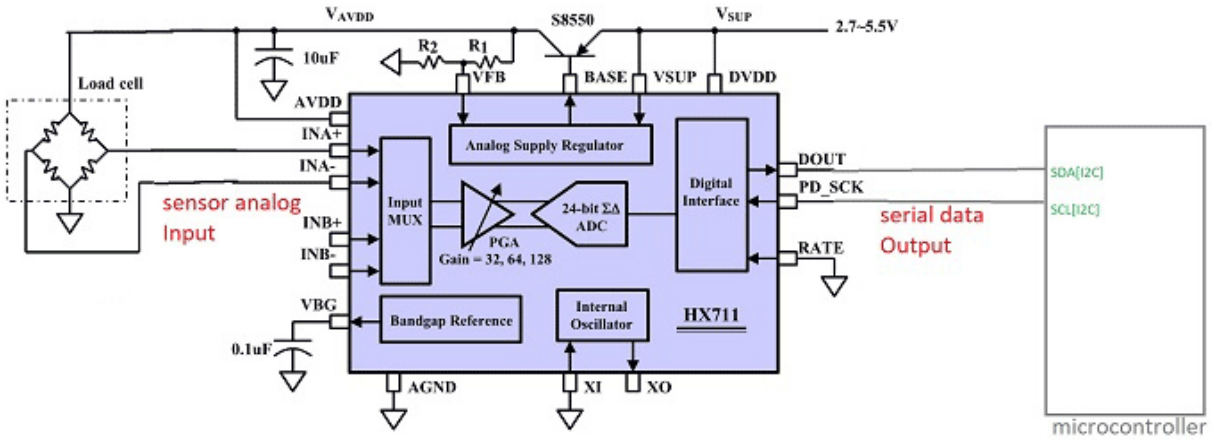


Image Reference: https://microcontrollerslab.com/wp-content/uploads/2020/03/weigh-scale-application-circuit-HX711.png

Physical Dimension & Power ratings:

- Differential input voltage: ±40mV (Full-scale differential input voltage is ± 40mV)
- Data accuracy: 24-bit (24-bit A / D converter chip.)
- Refresh frequency: 10/80 Hz
- Operating Voltage: 2.7V to 5VDC
- Operating current: <10 mA
- Size: 24x16mm

Type of Interface**:** After the analog signal is amplified through the actuator i.e. HX711 Load Cell Amplifier Module and upon conversion into digital signal it is sent to Raspberry Pi for making an interface of Load cell with Raspberry Pi. The interfacing method used over here is again physical interfacing.

Display Devices

The display devices used in this feature are LCD and Computer screens.

Physical dimensions of LCD: The outline size of the LCD is 80.0 x 36.0 mm and the VA size of 66.0 x 16.0 mm and the maximum thickness is 13.2 mm.

The power rating of LCD: The input voltage ranges from 85Vac to 264Vac, and the LCD driver draws around 1mA or 1.1mA.

Pin diagram of LCD:

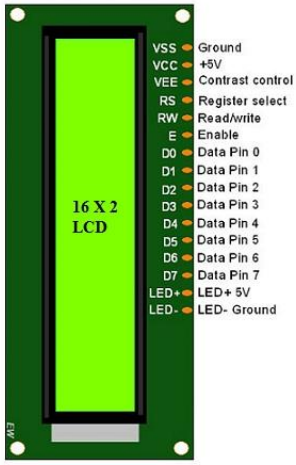


Figure: Pin Diagram of 16x2 LCD

Image Reference: https://www.elprocus.com/wp-content/uploads/lcd-16x2-pin-diagram.jpg

Type of Interfacing used for 16x2 LCD with Raspberry Pi: The LCD screen used to display weights at constant intervals uses physical interfacing with Raspberry Pi.

Physical dimensions of Computer Screen: In general, the size of a computer ranges from 19 to 34 inches when measured diagonally.
The power rating of the Computer Screen: The energy consumption of a desktop computer ranges from 60-250 watts.
Type of Interfacing used for Computer Screen with Raspberry Pi: The computer screen used to display email uses wireless interfacing with Raspberry Pi – Using the sSMTP Module.
A module called sSMTP is used to send emails from a local computer to a designated email host. It does not accept mail, extend aliases, or manage a queue because it is not a mail server (like the feature-rich mail server Sendmail). Its main function is to transfer automatic messages (such as system notifications) away from your computer and to an external email address.

Code to communicate with actuators and displays:

LED:

```
def begin():

  lcdcmd(0x33)
  lcdcmd(0x32)
  lcdcmd(0x06)
  lcdcmd(0x0C)
  lcdcmd(0x28)
  lcdcmd(0x01)
  time.sleep(0.0005)


def lcdcmd(ch):
  gpio.output(RS, 0)
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)

  if ch&0x10==0x10:
    gpio.output(D4, 1)

  if ch&0x20==0x20:
    gpio.output(D5, 1)

  if ch&0x40==0x40:
    gpio.output(D6, 1)

  if ch&0x80==0x80:
    gpio.output(D7, 1)

  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)

 # Low bits

  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)

  if ch&0x01==0x01:
    gpio.output(D4, 1)

  if ch&0x02==0x02:
    gpio.output(D5, 1)

  if ch&0x04==0x04:
    gpio.output(D6, 1)

  if ch&0x08==0x08:
    gpio.output(D7, 1)

  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)
```

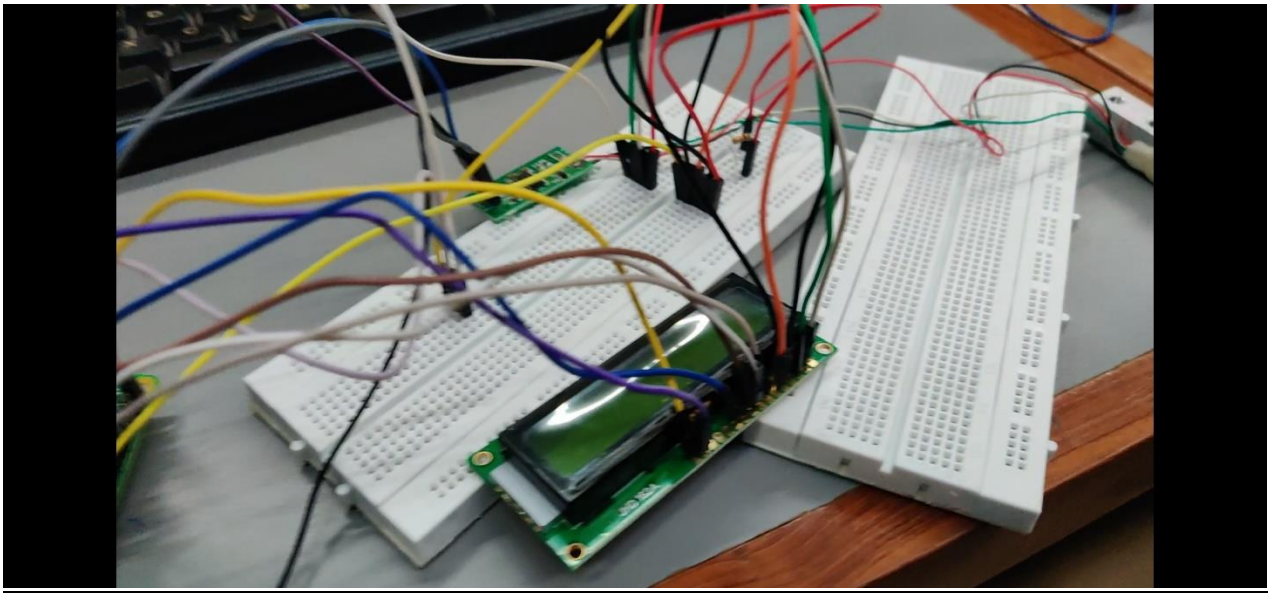
```
60  def lcdwrite(ch):
61
62    gpio.output(RS, 1)
63    gpio.output(D4, 0)
64    gpio.output(D5, 0)
65    gpio.output(D6, 0)
66    gpio.output(D7, 0)
67
68    if ch&0x10==0x10:
69      gpio.output(D4, 1)
70
71    if ch&0x20==0x20:
72      gpio.output(D5, 1)
73
74    if ch&0x40==0x40:
75      gpio.output(D6, 1)
76
77    if ch&0x80==0x80:
78      gpio.output(D7, 1)
79
80    gpio.output(EN, 1)
81    time.sleep(0.005)
82    gpio.output(EN, 0)
83
84
85    # Low bits
86
87    gpio.output(D4, 0)
88    gpio.output(D5, 0)
89    gpio.output(D6, 0)
90    gpio.output(D7, 0)
91
92    if ch&0x01==0x01:
93      gpio.output(D4, 1)
94
95    if ch&0x02==0x02:
96      gpio.output(D5, 1)
97
98    if ch&0x04==0x04:
99      gpio.output(D6, 1)
100
101   if ch&0x08==0x08:
102     gpio.output(D7, 1)
103
104   gpio.output(EN, 1)
105   time.sleep(0.005)
106   gpio.output(EN, 0)
107
108
109 def lcdclear():
110   lcdcmd(0x01)
111
112
```

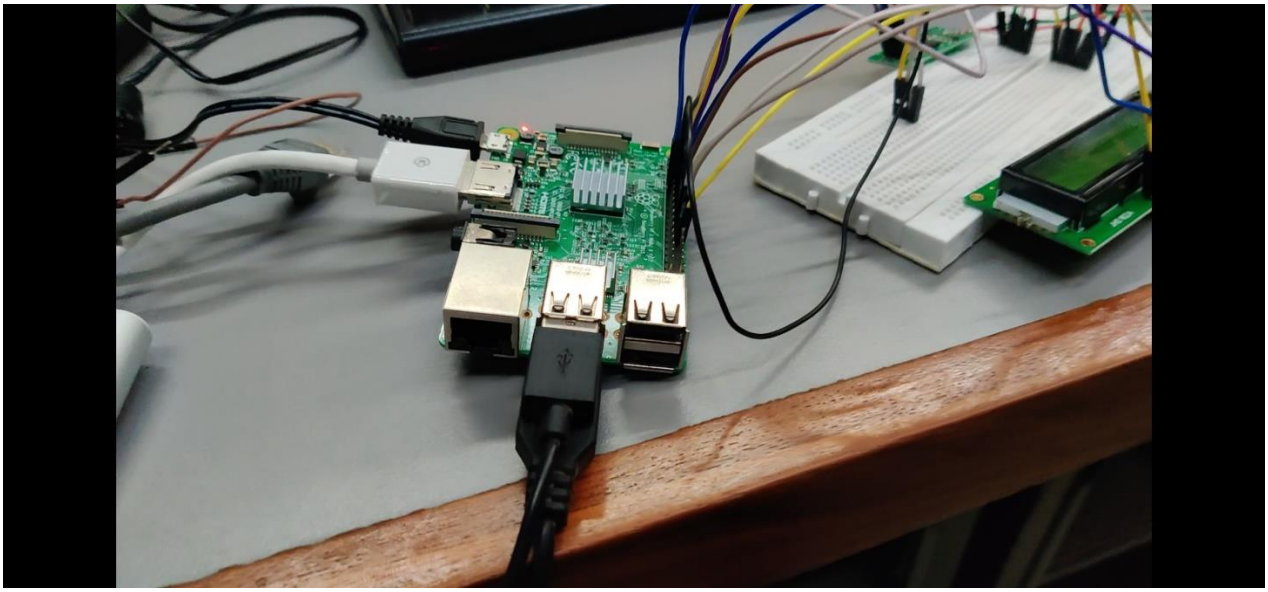
```
113  def lcdprint(Str):
114
115    l=0;
116    l=len(Str)
117
118    for i in range(l):
119      lcdwrite(ord(Str[i]))
120
121
122
123  def setCursor(x,y):
124
125      if y == 0:
126          n=128+x
127
128      elif y == 1:
129          n=192+x
130
131      lcdcmd(n)
132
```

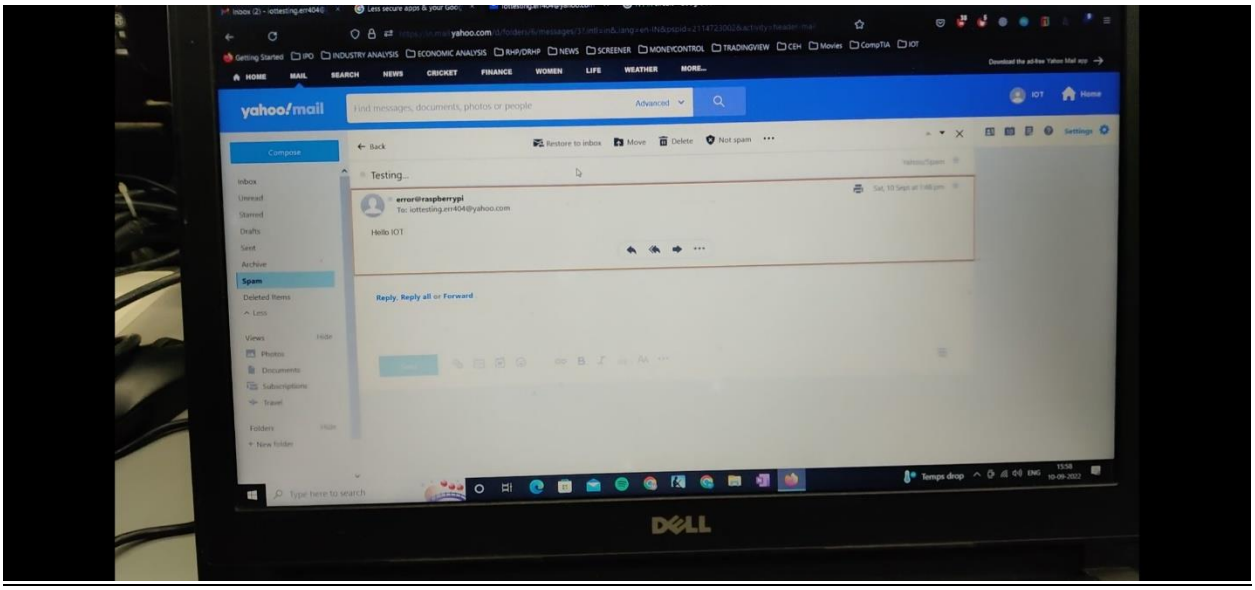
Send Mail:

```
def show_weight():

    count= readCount()
    w=0
    w=(count-sample)/106
    print(w,"g")
    setCursor(0,0)
    data=str(w);
    lcdprint(data)
    lcdprint("g   ")
    global flag

    if w>300:
        if flag == 0:
            lcdclear()
            lcdprint("Container Is Full")
            setCursor(0,1);
            lcdprint("Sending Email")
            server = smtplib.SMTP('smtp.mail.yahoo.com', 587)
            server.starttls()
            server.login("iottesting.err404@yahoo.com", "password")
            msg = "Alert.... Container Full"
            server.sendmail("iottesting.err404@yahoo.com", "shethpurvam@yahoo.com", msg)
            server.quit()
            lcdclear()
            lcdprint("Email Sent")
            flag=1;
            lcdclear()

    elif w<30:
      if flag==1:
        lcdclear()
        lcdprint("Container Empty")
        setCursor(0,1);
        lcdprint("Sending Email")
        server = smtplib.SMTP(''smtp.mail.yahoo.com', 587)
        server.starttls()
        server.login("iottesting.err404@yahoo.com", "password")
        msg = "Alert.... Container is Empty"
        server.sendmail("iottesting.err404@yahoo.com", "shethpurvam@yahoo.com", msg)
        server.quit()
        lcdclear()
        lcdprint("Email Sent")
        flag=0;
        lcdclear()

    return jsonify(result=w)

```

Photos of working hardware:

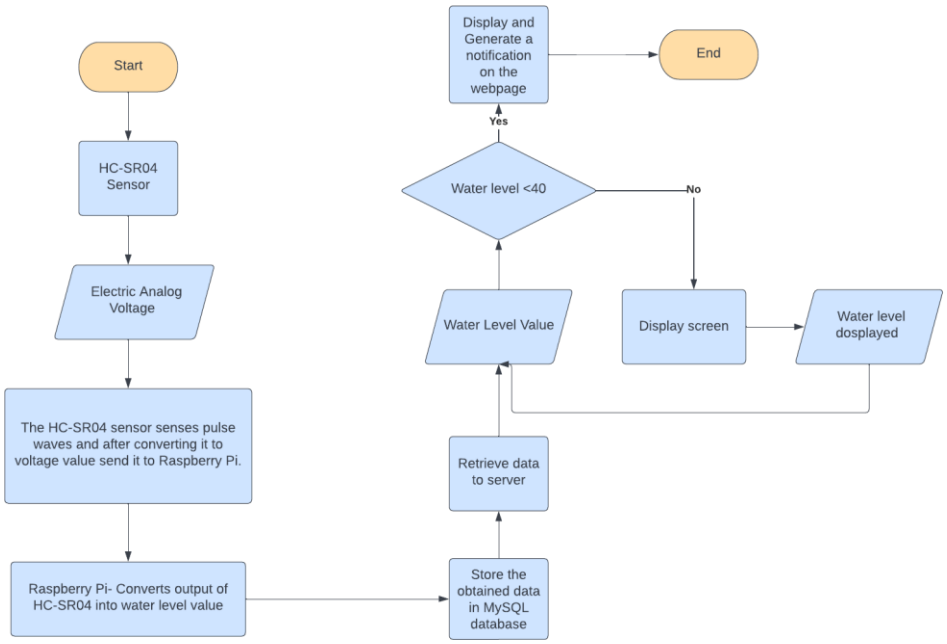





yahoo/mail
Find messages, documents, photos or people
Advanced
Compose
Inbox
Unread
Starred
Drafts
Sent
Archive
Spam
Deleted Items
Back
Restore to inbox
Move
Delete
Not spam
Testing...
error@raspberrypi
To: iottesting.err404@yahoo.com
Hello IOT
Reply, Reply all or Forward
DELL

Project Report -3

Flowchart of Process



Details of Sensors

HC-SR04 Sensor:

Operating Principle of HC-SR04:

The HC SR04 sensor works on the SONAR and RADAR principles that are used to determine the distance of an object. This sensor generates high-frequency sound waves. When this sound hits an object, they reflect an echo, which the receiver senses.

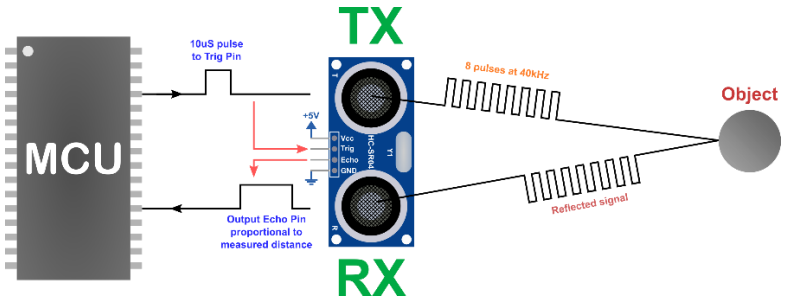


Figure: Operating principle of HC SR04 Sensor

Image Reference: https://trionprojects.org/wp-content/uploads/2020/02/how-ultrasonic-sensor-works-1200x446.png

Physical Dimension:

- Length: 43mm
- Width: 20mm
- Height: 15mm
- Centre screw hole dimension: 40mm × 15mm
- Screw hole diameter: 1mm(M1)
- Transmitter diameter: 8mm

Power Rating:

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 0V (Low)

Pin Diagram:

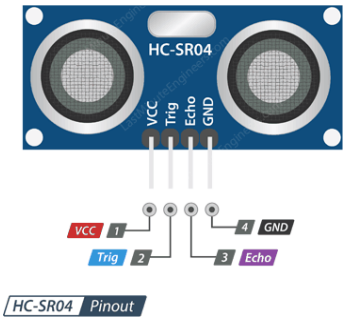
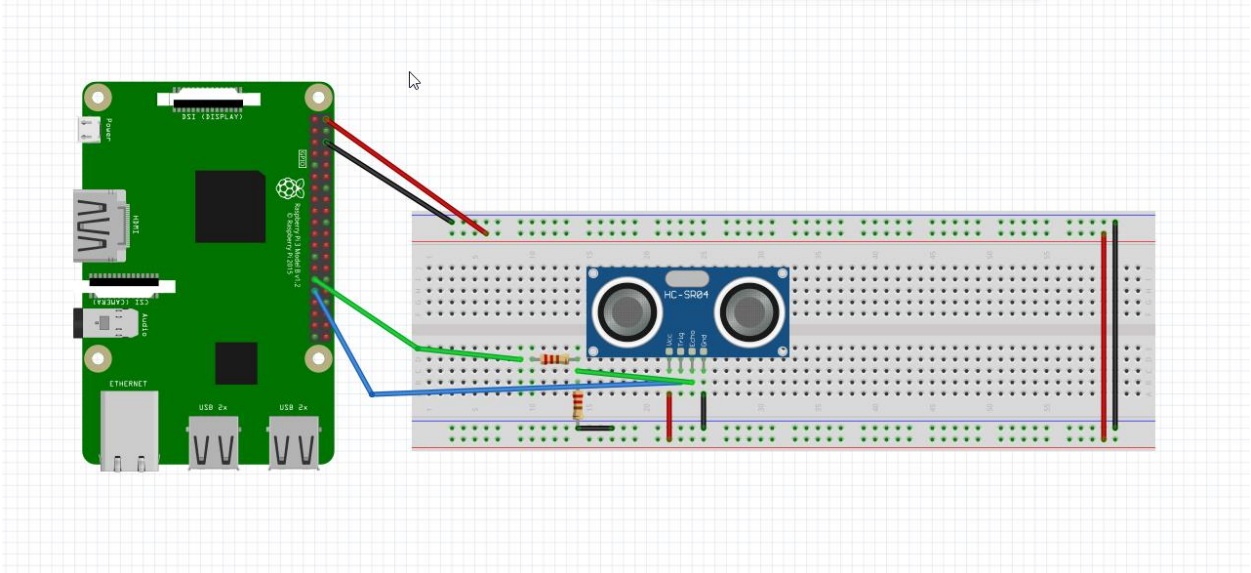


Figure: Pin Diagram of HC SR04

Image Reference: https://www.utilcell.com/upload/img1.jpg

Type of Interface:



To make an interface with Raspberry Pi, the HC-SR04 is connected to VCC to Pin 2, GND is connected to Pin 6, TRIG is connected to Pin 12 i.e., GPIO18, and ECHO is connected to Pin 18 i.e., GPIO24.

Code to communicate with sensors:

Sensor.py file

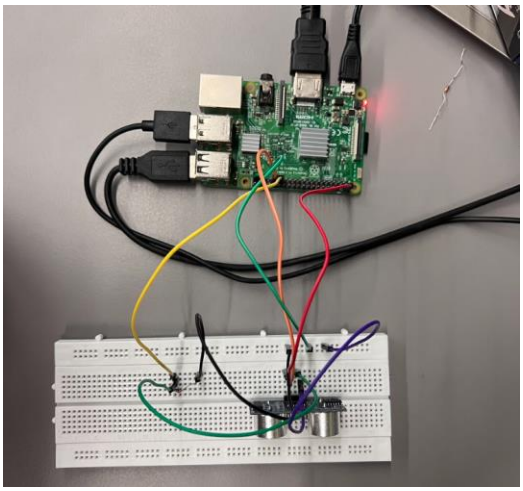
```
import RPi.GPIO as GPIO
import time,os
import datetime

TRIG = 6
ECHO = 5
ALARM = 23

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG, False)

GPIO.setup(ALARM,GPIO.OUT)
GPIO.output(ALARM, True)

print ("Waiting For Sensor To Settle")
time.sleep(1)

def get_distance():
    dist_add = 0
    for x in range(20):
        try:
            GPIO.output(TRIG, True)
            time.sleep(0.00001)
            GPIO.output(TRIG, False)

            while GPIO.input(ECHO)==0:
                pulse_start = time.time()

            while GPIO.input(ECHO)==1:
                pulse_end = time.time()

            pulse_duration = pulse_end - pulse_start

            distance = pulse_duration * 17150

            distance = round(distance, 3)
            print (x, "distance: ", distance)

            dist_add = dist_add + distance

            time.sleep(.1)

        except Exception as e:

            pass

    avg_dist=dist_add/20
    dist=round(avg_dist,3)

    return dist

def sendData_to_remoteServer(dist):
```

```
def main():

    distance=get_distance()

    print ("distance: ", distance)
    sendData_to_remoteServer(distance)
    low_level_warning(distance)
    print ("--------------------")

if __name__ == '__main__':
    main()
```

Setup_cron.sh file

```
FILE_NAME="sensor.py"

crontab -u $USER -l| grep $PWD/$FILE_NAME > /dev/null
if [ $? -eq 0 ]
then
    echo "$FILE_NAME is already running"
else
    crontab -l > mycron
    echo "* * * * * sudo python $PWD/$FILE_NAME &" >> mycron
    crontab mycron
    rm mycron
    echo "cron task added. Now $PWD/$FILE_NAME file will run every minute"
fi
```

Photos of working hardware:



Details of Actuators & Display Devices

Display Devices:

The display devices used in this feature is Computer screen which is used to display the amount of water that is currently present in the container.

Physical Dimension: In general, the size of a computer ranges from 19 to 34 inches when measured diagonally.

Power Rating: The energy consumption of a desktop computer ranges from 60-250 watts

Type of Interfacing used for Computer Screen with Raspberry Pi: In this interfacing, the data is sent to the display screen from the Raspberry Pi via the IP address of the target display screen where the data is supposed to be transmitted.

Python Code to communicate with display:

From sensor.py

```python
def sendData_to_remoteServer(dist):

    url_remote="http://192.168.172.92/water-tank/insert_data.php?dist=" + str(dist)
    cmd="curl -s " + url_remote
    result=os.popen(cmd).read()
    print (cmd)

def low_level_warning(dist):
    tank_height=114
    level=tank_height-dist
    if(level<40):
        print("level low : ", level)
        GPIO.output(ALARM, False)
    else:
        GPIO.output(ALARM, True)
        print("level ok")

def main():

    distance=get_distance()

    print ("distance: ", distance)
    sendData_to_remoteServer(distance)
    low_level_warning(distance)
    print ("----------------------")

if __name__ == '__main__':
    main()
```

index.html file

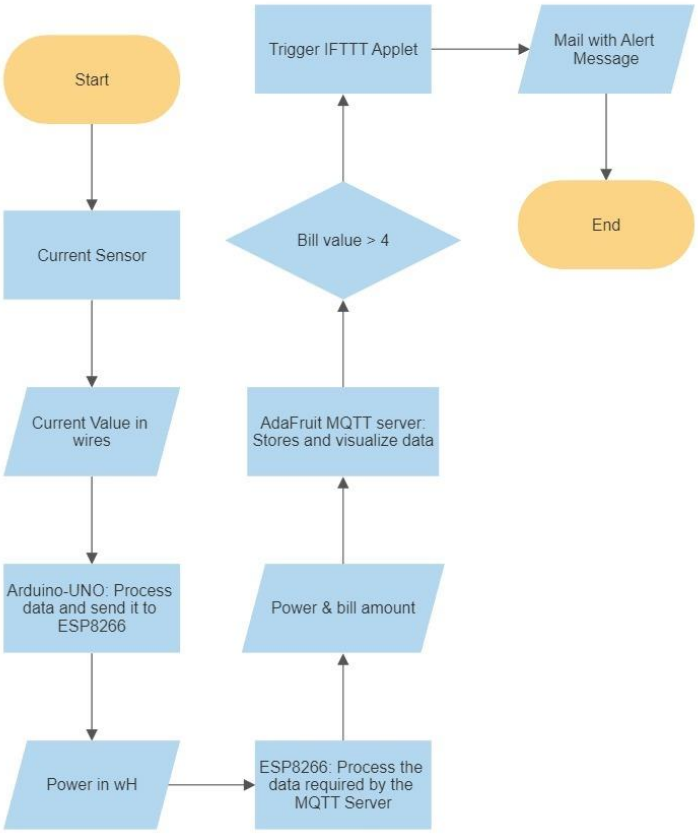
```html
<!doctype html>
<html lang="en">

<head>
<script type="text/javascript" src='jquery.min.js'></script>

<script>
    function data_request_timer(){
        window.setInterval(get_data, 2000);
    }

    function get_data(){
        console.log("fetching data from server");

        $.get("read_data1.php",
        function(data, status){
            document.getElementById("info").innerHTML = data;
        });

    }

</script>
</head>

<body onload="data_request_timer()">

<div style='border:0px solid blue; padding:0%;'>
    <p id='info'></p>
</div>

<button>
    <a style='text-decoration:none;font-size:12px' href='../insert_data.php?dist=65' target='_blank'>Insert dummy value</a>
</button>

</body>
</html>
```

insert_data.php file

```php
<?php
date_default_timezone_set("Asia/Kolkata");

include_once 'util.php';

$val=$_GET["dist"];

insert_in_db($val);

function insert_in_db($val){
   echo"<p>Inserting value <b>$val</b> in database</p>";

    $connection=create_db_connection();

    $t=time();
    $date_time= date('Y-m-d H:i:s', $t);

    echo "<p>$date_time</p>";

    $sql="INSERT INTO `level_log`(`level`, `timestamp`, `date_time`)
                    VALUES ('$val','$t', '$date_time')";

    $result = mysqli_query($connection,$sql);

        if ( !$result ) {
          die(mysql_error()."\n".$sql);
          echo" !!!<br> ";
        }

       if ( $result ) {
          echo" <p>successfully inserted >>>> </p> ";
      }

    close_db_connection($connection);

}
?>
```

Photos of working hardware:

Project Report-4
Flowchart of Process



Details of Sensors

ACS712 Current Sensor:

Operating Principle of ACS712:

The hall effect sensor generates a magnetic field that allows it to detect incoming currents. After being activated, the hall effect sensor produces a voltage proportionate to its magnetic field, which is utilized to determine the current flow.

Figure: Operating principle of ACS712 Sensor

Image Reference: https://i.stack.imgur.com/Noni1.png

Physical Dimension:

- Length: 31mm
- Width: 13mm
- Height: 14mm
- Weight: 3gm

Power Rating:

- Supply Voltage: 4.5V ~ 5.5V DC
- Measure Current Range: -20A ~ 20A
- Sensitivity: 100mV/A

Pin Diagram:

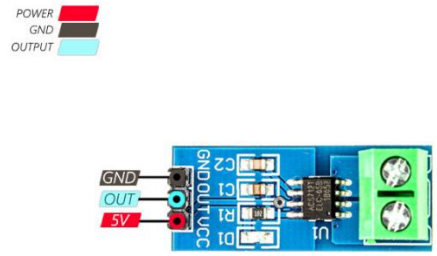
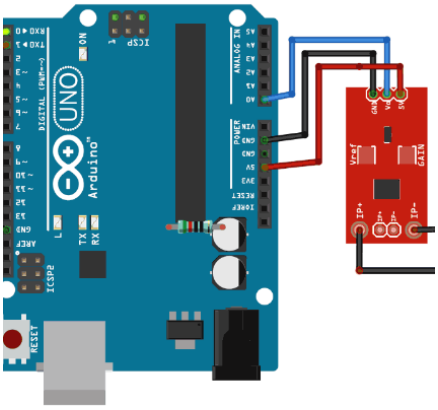


Figure: Pin Diagram of ACS712

Image Reference: https://electropeak.com/learn/wp-content/uploads/2020/09/ACS712-Arduino-Pinout.jpg

Type of Interface:



To make an interface with Arduino, the ACS712 is connected to VCC to Pin 8, GND is connected to Pin 5, A0 is connected to Pin 7, and Pins 1 and 4 are connected to any AC appliance.

Code to communicate with sensor:

Sensor | Arduino 1.8.9

File Edit Sketch Tools Help

Sensor

```
#include "ACS712.h"

char watt[5];

ACS712 sensor(ACS712_30A, A0);

unsigned long last_time =0;

unsigned long current_time =0;

float Wh =0 ;

void setup() {

  Serial.begin(115200);

  sensor.calibrate();

}


void loop() {

  float V = 230;

  float I = sensor.getCurrentAC();

// Serial.println(I);
```

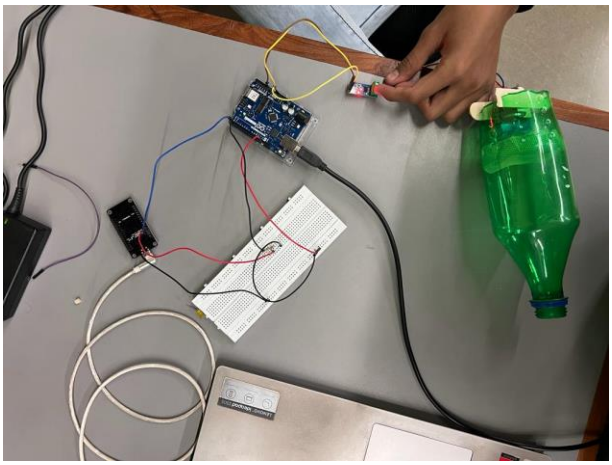
Sensor | Arduino 1.8.9

File Edit Sketch Tools Help

Sensor

```
  sensor.calibrate();

}


void loop() {

  float V = 230;

  float I = sensor.getCurrentAC();

// Serial.println(I);

  float P = V * I;

  last_time = current_time;

  current_time = millis();

  Wh = Wh+  P *(( current_time -last_time) /3600000.0) ;

   dtostrf(Wh, 4, 2, watt);

Serial.write(watt);

  delay(10000);

}
```

Photos of working hardware:



Details of Actuators & Display Devices

ESP8266 WiFi Module:

Operating Principle of ESP8266 WiFi Module:

A self-contained SOC with an integrated TCP/IP protocol stack, the ESP8266 WiFi Module allows any microcontroller to access your WiFi network. The ESP8266 is capable of offloading all WiFi networking tasks from another application processor or hosting an application.

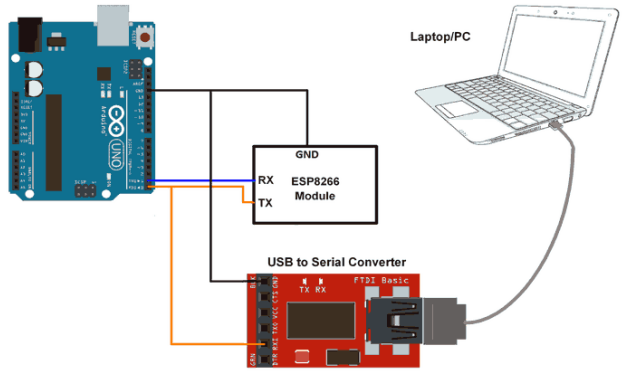


Image Reference: https://www.electronicwings.com/storage/PlatformSection/TopicContent/308/description/response%20image.png

Physical Dimension & Power ratings:

- Operating Voltage: 3.3V
- Input Voltage: 4.5V ~ 10V
- ADC Range: 0V ~ 3.3V
- Size: 49x26mm

Pin Diagram:

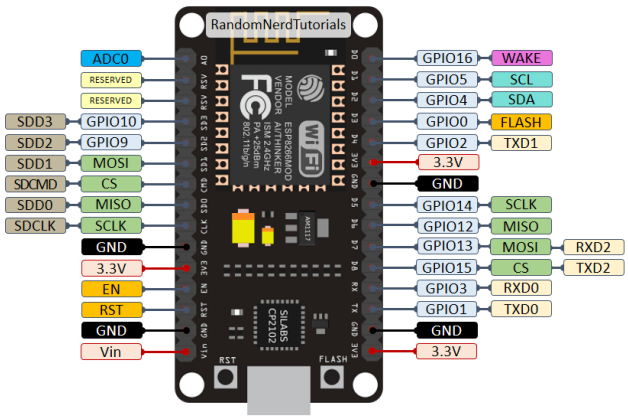
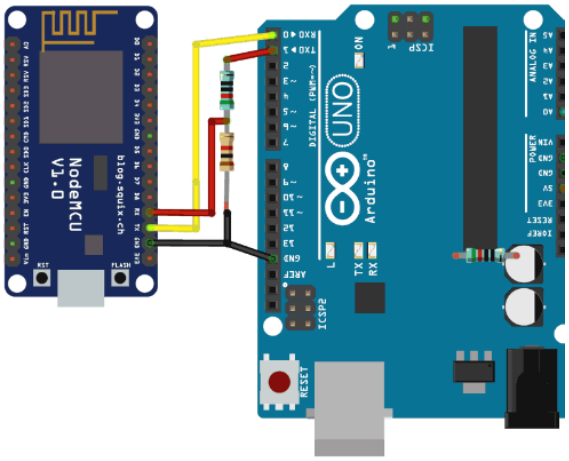


Image Reference: https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/05/ESP8266-NodeMCU-kit-12-E-pinout-gpio-pin.png?w=817&quality=100&strip=all&ssl=1

Type of Interface:



To make an interface with Arduino, the ESP8266 is connected to RX0 to GPIO1, and TX0 is connected to GPIO3 and Arduino's GND with NodeMCU's GND.

Display Devices:

The display devices used in this feature is Computer screen which is used to display the amount of water that is currently present in the container.

Physical Dimension: In general, the size of a computer ranges from 19 to 34 inches when measured diagonally.

Power Rating: The energy consumption of a desktop computer ranges from 60-250 watts

Type of Interfacing used for Computer Screen with ESP8266: Here, wireless interfacing is used. In this interfacing it will send data from ESP8266 to MQTT, which helps in visualizing and storing of data.

Code to communicate with actuator:

mqtt_esp8266

```
/*************************************************
  Adafruit MQTT Library ESP8266 Example

  Must use ESP8266 Arduino from:
    https://github.com/esp8266/Arduino

  Works great with Adafruit's Huzzah ESP board & Feather
  ----> https://www.adafruit.com/product/2471
  ----> https://www.adafruit.com/products/2821



  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Tony DiCola for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 *************************************************/
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "Raj's iphone"
#define WLAN_PASS       "12345678"
char watt[5];

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
```

mqtt_esp8266

```
#define AIO_USERNAME    "iottesting_err404"
#define AIO_KEY         "aio_Hflh08LPcL6U5NwNQ52IZzrd77Bl"

/************ Global State (you don't need to change this!) ******************/

WiFiClient client;

int bill_amount = 0;

unsigned int energyTariff = 8.0;

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Publish Power = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/Power");

Adafruit_MQTT_Publish bill = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/bill");

void MQTT_connect();


void setup() {

  Serial.begin(115200);

  delay(10);

  Serial.println(F("Adafruit MQTT demo"));


  // Connect to WiFi access point.

  Serial.println(); Serial.println();
```

mqtt_esp8266

```
  Serial.print("Connecting to ");

  Serial.println(WLAN_SSID);


  WiFi.begin(WLAN_SSID, WLAN_PASS);

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println();


  Serial.println("WiFi connected");

  Serial.println("IP address: "); Serial.println(WiFi.localIP());

}


void loop() {

  // Ensure the connection to the MQTT server is alive (this will make the first

  // connection and automatically reconnect when disconnected).  See the MQTT_connect

  // function definition further below.
```

mqtt_esp8266

```
 MQTT_connect();


int i=0;

float watt1;

if(Serial.available() > 0 ){

  delay(100); //allows all serial sent to be received together

   while(Serial.available() && i<5) {

    watt[i++] = Serial.read();

   }

   watt[i++]='\0';

 }


watt1 = atof(watt);

 bill_amount = watt1 * (energyTariff/1000);      // 1unit = 1kwH

 Serial.print(F("\nSending Power val "));

 Serial.println(watt1);

 Serial.print("...");
```

mqtt_esp8266

```
  if (! Power.publish(watt1)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

   if (! bill.publish(bill_amount)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

if (bill_amount==4){
for (int i =0; i<=2; i++)
{
  bill.publish(bill_amount);
delay(5000);
}
```

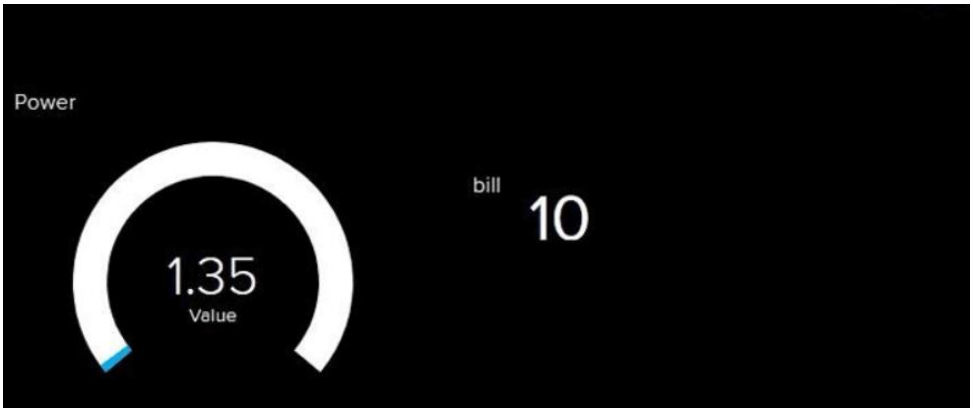
mqtt_esp8266

```
bill_amount =6;

}


delay(5000);

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT_connect() {

  int8_t ret;


  // Stop if already connected.

  if (mqtt.connected()) {

    return;

  }


  Serial.print("Connecting to MQTT... ");
```

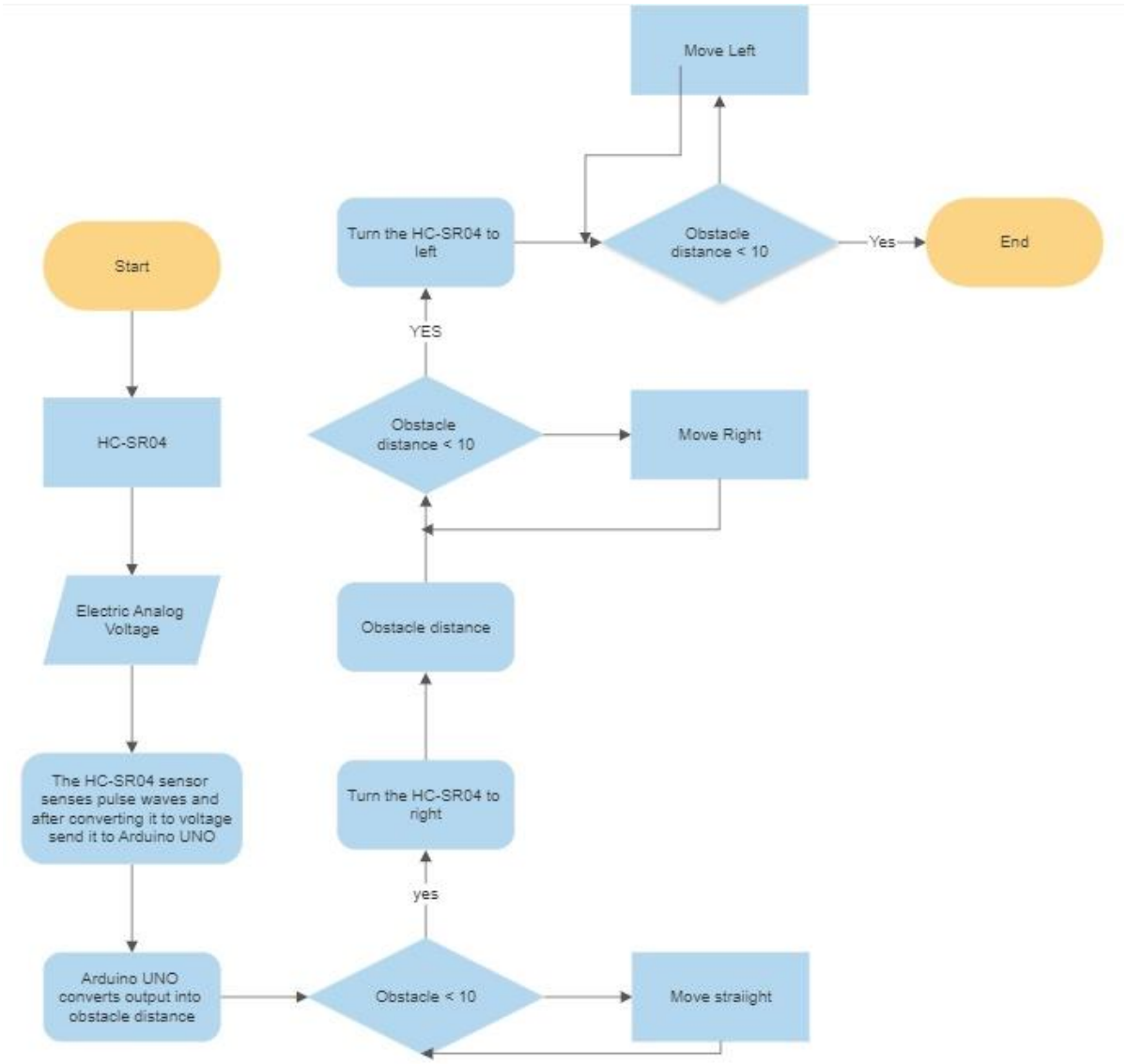
mqtt_esp8266

```
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```

Photos of working hardware:



Flowchart of the program:



Details of Sensors:

HC-SR04 Sensor:

Operating Principle of HC-SR04:

The HC SR04 sensor works on the SONAR and RADAR principles that are used to determine the distance of an object. This sensor generates high-frequency sound waves. When this sound hit an object they reflect an echo, which is sensed by the receiver.

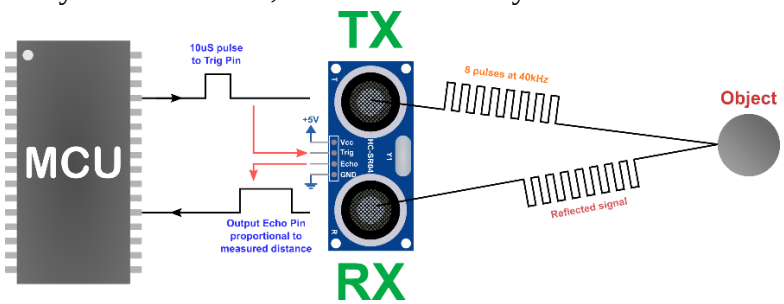


Figure: Operating principle of HC SR04 Sensor

Image Reference: https://trionprojects.org/wp-content/uploads/2020/02/how-ultrasonic-sensor-works-1200x446.png

Physical Dimension:

- Length: 43mm
- Width: 20mm
- Height: 15mm
- Centre screw hole dimension: 40mm × 15mm
- Screw hole diameter: 1mm(M1)
- Transmitter diameter: 8mm

Power Rating:

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 0V (Low)

Pin Diagram:

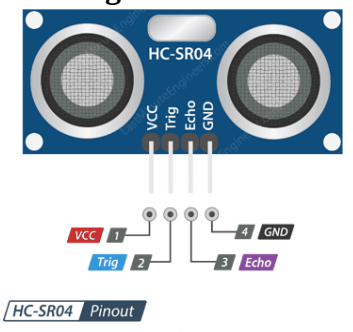
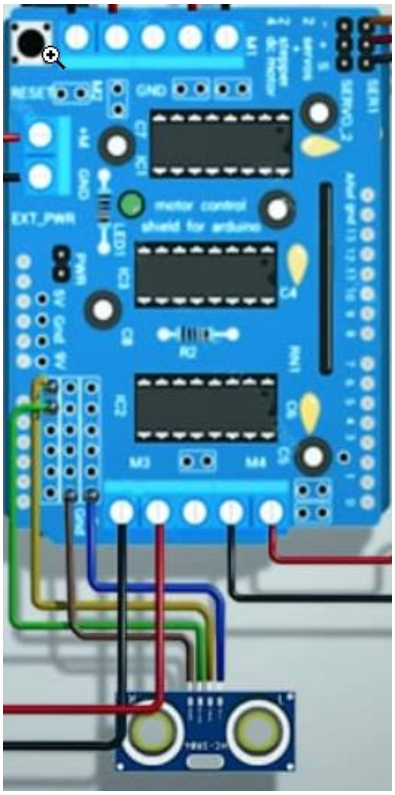


Figure: Pin Diagram of HC SR04

Image Reference: https://www.utilcell.com/upload/img1.jpg

Type of Interface:



To make interface with Arduino motor shield, connect 5V pin of sensor to 5V of motor shield, connect sensor's GND to GND of motor shield.

Code to communicate with sensors:

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
// ---------------------------------------------------------------------------
// Created by Tim Eckel - teckel@leethost.com
// Copyright 2012 License: GNU GPL v3 http://www.gnu.org/licenses/gpl-3.0.html
//
// See "NewPing.h" for purpose, syntax, version history, links, and more.
// ---------------------------------------------------------------------------

#include "NewPing.h"


// ---------------------------------------------------------------------------
// NewPing constructor
// ---------------------------------------------------------------------------

NewPing::NewPing(uint8_t trigger_pin, uint8_t echo_pin, int max_cm_distance) {
    _triggerBit = digitalPinToBitMask(trigger_pin); // Get the port register bitmask for the trigger pin.
    _echoBit = digitalPinToBitMask(echo_pin);       // Get the port register bitmask for the echo pin.

    _triggerOutput = portOutputRegister(digitalPinToPort(trigger_pin)); // Get the output port register for the trigger pin.
    _echoInput = portInputRegister(digitalPinToPort(echo_pin));         // Get the input port register for the echo pin.

    _triggerMode = (uint8_t *) portModeRegister(digitalPinToPort(trigger_pin)); // Get the port mode register for the trigger pin.

    _maxEchoTime = min(max_cm_distance, MAX_SENSOR_DISTANCE) * US_ROUNDTRIP_CM + (US_ROUNDTRIP_CM / 2); // Calculate the maximum distance in uS.

#if DISABLE_ONE_PIN == true
    *_triggerMode |= _triggerBit; // Set trigger pin to output.
#endif
}


// ---------------------------------------------------------------------------
// Standard ping methods
// ---------------------------------------------------------------------------

unsigned int NewPing::ping() {
    if (!ping_trigger()) return NO_ECHO;               // Trigger a ping, if it returns false, return NO_ECHO to the calling function.
    while (*_echoInput & _echoBit)                     // Wait for the ping echo.
        if (micros() > _max_time) return NO_ECHO;      // Stop the loop and return NO_ECHO (false) if we're beyond the set maximum distance.
    return (micros() - (_max_time - _maxEchoTime) - 5); // Calculate ping time, 5uS of overhead.
}


unsigned int NewPing::ping_in() {
    unsigned int echoTime = NewPing::ping();          // Calls the ping method and returns with the ping echo distance in uS.
    return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
}
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
    return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
}


unsigned int NewPing::ping_cm() {
    unsigned int echoTime = NewPing::ping();          // Calls the ping method and returns with the ping echo distance in uS.
    return NewPingConvert(echoTime, US_ROUNDTRIP_CM); // Convert uS to centimeters.
}


unsigned int NewPing::ping_median(uint8_t it) {
    unsigned int uS[it], last;
    uint8_t j, i = 0;
    uS[0] = NO_ECHO;
    while (i < it) {
        last = ping();           // Send ping.
        if (last == NO_ECHO) {   // Ping out of range.
            it--;                // Skip, don't include as part of median.
            last = _maxEchoTime; // Adjust "last" variable so delay is correct length.
        } else {                 // Ping in range, include as part of median.
            if (i > 0) {         // Don't start sort till second ping.
                for (j = i; j > 0 && uS[j - 1] < last; j--) // Insertion sort loop.
                    uS[j] = uS[j - 1]; // Shift ping array to correct position for sort insertion.
            } else j = 0;        // First ping is starting point for sort.
            uS[j] = last;        // Add last ping to array in sorted position.
            i++;                 // Move to next ping.
        }
        if (i < it) delay(PING_MEDIAN_DELAY - (last >> 10)); // Millisecond delay between pings.
    }
    return (uS[it >> 1]); // Return the ping distance median.
}


// ---------------------------------------------------------------------------
// Standard ping method support functions (not called directly)
// ---------------------------------------------------------------------------

boolean NewPing::ping_trigger() {
#if DISABLE_ONE_PIN != true
    *_triggerMode |= _triggerBit;    // Set trigger pin to output.
#endif
    *_triggerOutput &= ~_triggerBit; // Set the trigger pin low, should already be low, but this will make sure it is.
    delayMicroseconds(4);            // Wait for pin to go low, testing shows it needs 4uS to work every time.
    *_triggerOutput |= _triggerBit;  // Set trigger pin high, this tells the sensor to send out a ping.
    delayMicroseconds(10);           // Wait long enough for the sensor to realize the trigger pin is high. Sensor specs say to wait 10uS.
    *_triggerOutput &= ~_triggerBit; // Set trigger pin back to low.
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
#if DISABLE_ONE_PIN != true
    *_triggerMode &= ~_triggerBit;   // Set trigger pin to input (when using one Arduino pin this is technically setting the echo pin to input as both are tied to the
#endif

    _max_time =  micros() + MAX_SENSOR_DELAY;                // Set a timeout for the ping to trigger.
    while (*_echoInput & _echoBit && micros() <= _max_time) {} // Wait for echo pin to clear.
    while (!(*_echoInput & _echoBit))                        // Wait for ping to start.
        if (micros() > _max_time) return false;              // Something went wrong, abort.

    _max_time = micros() + _maxEchoTime; // Ping started, set the timeout.
    return true;                         // Ping started successfully.
}


#ifdef ENABLE_TIMER_BASED_PING

// ---------------------------------------------------------------------------
// Timer interrupt ping methods (won't work with ATmega8 and ATmega128)
// ---------------------------------------------------------------------------

void NewPing::ping_timer(void (*userFunc)(void)) {
    if (!ping_trigger()) return;         // Trigger a ping, if it returns false, return without starting the echo timer.
    timer_us(ECHO_TIMER_FREQ, userFunc); // Set ping echo timer check every ECHO_TIMER_FREQ uS.
}


boolean NewPing::check_timer() {
    if (micros() > _max_time) { // Outside the timeout limit.
        timer_stop();           // Disable timer interrupt
        return false;           // Cancel ping timer.
    }

    if (!(*_echoInput & _echoBit)) { // Ping echo received.
        timer_stop();                // Disable timer interrupt
        ping_result = (micros() - (_max_time - _maxEchoTime) - 13); // Calculate ping time, 13uS of overhead.
        return true;                 // Return ping echo true.
    }

    return false; // Return false because there's no ping echo yet.
}


// ---------------------------------------------------------------------------
// Timer2/Timer4 interrupt methods (can be used for non-ultrasonic needs)
// ---------------------------------------------------------------------------
```

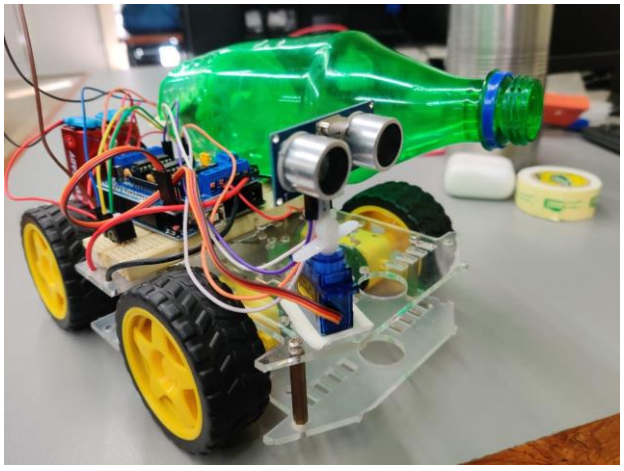
D: > Study > SEM 7 > IoT > Project > Car_sensor

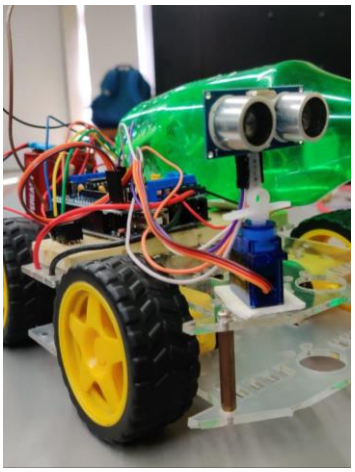
```
// Variables used for timer functions
void (*intFunc)();
void (*intFunc2)();
unsigned long _ms_cnt_reset;
volatile unsigned long _ms_cnt;


void NewPing::timer_us(unsigned int frequency, void (*userFunc)(void)) {
    timer_setup();      // Configure the timer interrupt.
    intFunc = userFunc; // User's function to call when there's a timer event.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    OCR4C = min((frequency>>2) - 1, 255); // Every count is 4uS, so divide by 4 (bitwise shift right 2) subtract one, then make sure we don't go over 255 limit.
    TIMSK4 = (1<<TOIE4);                  // Enable Timer4 interrupt.
#else
    OCR2A = min((frequency>>2) - 1, 255); // Every count is 4uS, so divide by 4 (bitwise shift right 2) subtract one, then make sure we don't go over 255 limit.
    TIMSK2 |= (1<<OCIE2A);                // Enable Timer2 interrupt.
#endif
}


void NewPing::timer_ms(unsigned long frequency, void (*userFunc)(void)) {
    timer_setup();                       // Configure the timer interrupt.
    intFunc = NewPing::timer_ms_cntdwn;  // Timer events are sent here once every ms till user's frequency is reached.
    intFunc2 = userFunc;                 // User's function to call when user's frequency is reached.
    _ms_cnt = _ms_cnt_reset = frequency; // Current ms counter and reset value.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    OCR4C = 249;         // Every count is 4uS, so 1ms = 250 counts - 1.
    TIMSK4 = (1<<TOIE4); // Enable Timer4 interrupt.
#else
    OCR2A = 249;           // Every count is 4uS, so 1ms = 250 counts - 1.
    TIMSK2 |= (1<<OCIE2A); // Enable Timer2 interrupt.
#endif
}


void NewPing::timer_stop() { // Disable timer interrupt.
#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    TIMSK4 = 0;
#else
    TIMSK2 &= ~(1<<OCIE2A);
#endif
}
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
188  void NewPing::timer_setup() {
189  #if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
190      timer_stop(); // Disable Timer4 interrupt.
191      TCCR4A = TCCR4C = TCCR4D = TCCR4E = 0;
192      TCCR4B = (1<<CS42) | (1<<CS41) | (1<<CS40) | (1<<PSR4); // Set Timer4 prescaler to 64 (4uS/count, 4uS-1020uS range).
193      TIFR4 = (1<<TOV4);
194      TCNT4 = 0;    // Reset Timer4 counter.
195  #else
196      timer_stop();           // Disable Timer2 interrupt.
197      ASSR &= ~(1<<AS2);      // Set clock, not pin.
198      TCCR2A = (1<<WGM21);    // Set Timer2 to CTC mode.
199      TCCR2B = (1<<CS22);     // Set Timer2 prescaler to 64 (4uS/count, 4uS-1020uS range).
200      TCNT2 = 0;              // Reset Timer2 counter.
201  #endif
202  }
203
204
205  void NewPing::timer_ms_cntdwn() {
206      if (!_ms_cnt--) {           // Count down till we reach zero.
207          intFunc2();             // Scheduled time reached, run the main timer event function.
208          _ms_cnt = _ms_cnt_reset; // Reset the ms timer.
209      }
210  }
211
212
213  #if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
214  ISR(TIMER4_OVF_vect) {
215  #else
216  ISR(TIMER2_COMPA_vect) {
217  #endif
218      if(intFunc) intFunc(); // If wrapped function is set, call it.
219  }
220
221  #endif // ENABLE_TIMER_BASED_PING
222
223
224  // ---------------------------------------------------------------------------
225  // Conversion methods (rounds result to nearest inch or cm).
226  // ---------------------------------------------------------------------------
227
228  unsigned int NewPing::convert_in(unsigned int echoTime) {
229      return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
230  }
231
232
233  unsigned int NewPing::convert_cm(unsigned int echoTime) {
```

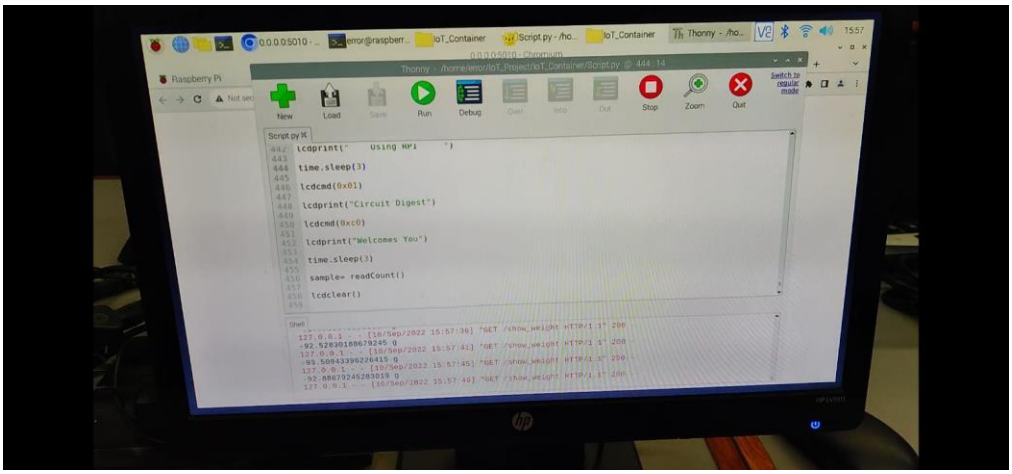
Photos of working hardware:



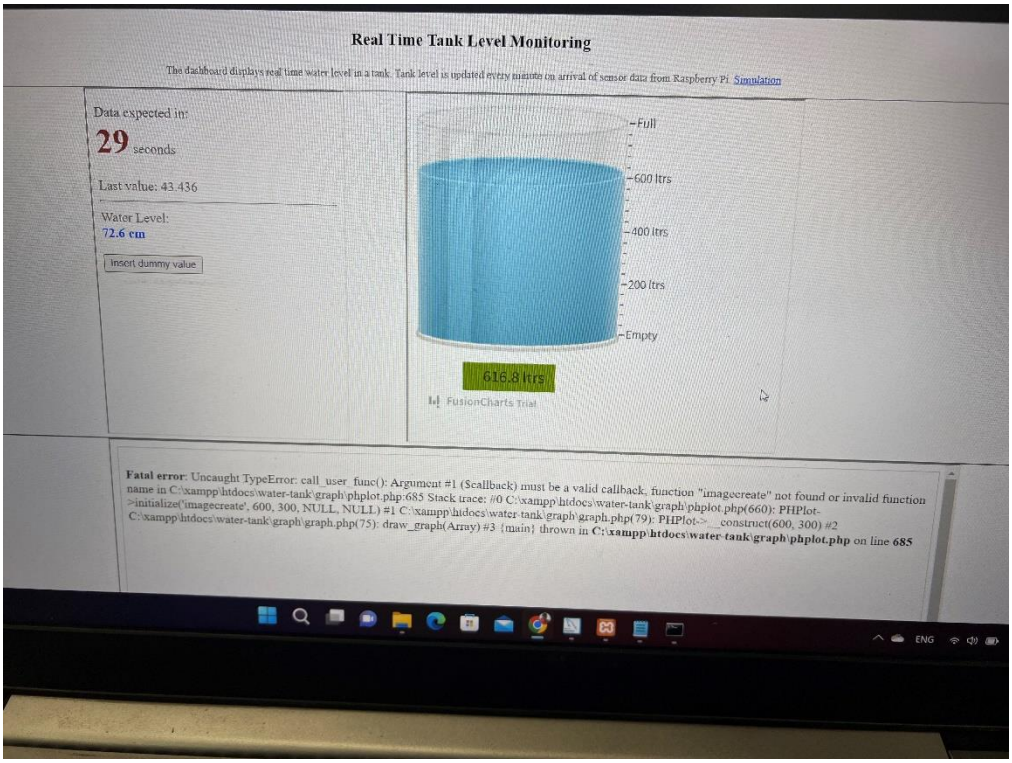


Details of the website used:

For Smart Weight Container:



For Smart Water Tank:



Code for the website:

```html
<!doctype html>
<html lang="en">

<head>
<script type="text/javascript" src='jquery.min.js'></script>

<script>
    function data_request_timer(){
        window.setInterval(get_data, 2000);
    }

    function get_data(){
        console.log("fetching data from server");

        $.get("read_data1.php",
        function(data, status){
            document.getElementById("info").innerHTML = data;
        });

    }

</script>
</head>

<body onload="data_request_timer()">

<div style='border:0px solid blue; padding:0%;'>
    <p id='info'></p>
</div>

<button>
    <a style='text-decoration:none;font-size:12px' href='../insert_data.php?dist=65' target='_blank'>Insert dummy value</a>
</button>

</body>
</html>
```

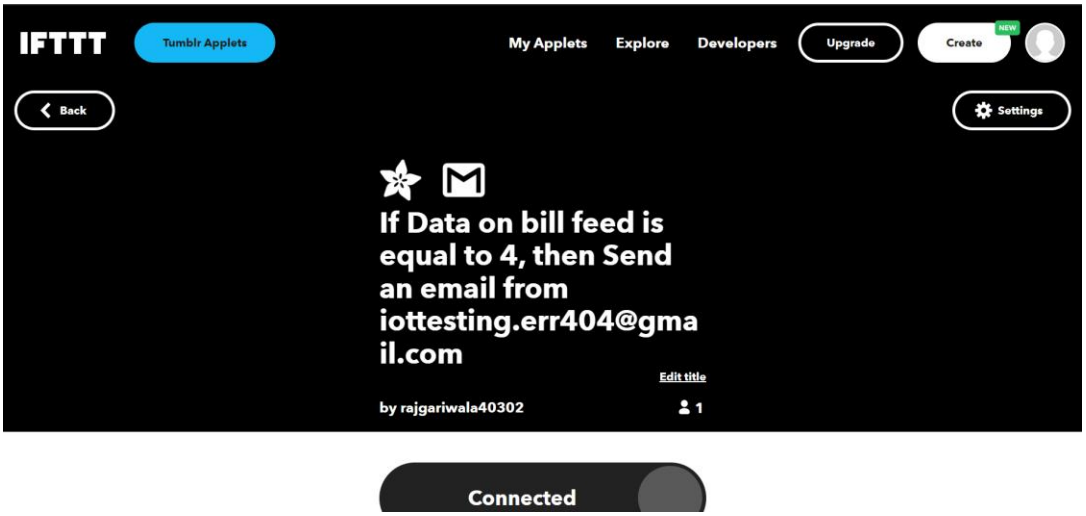
D: > Raj Study > 7th Sem > ECE504 > Project > Smart-Water-Tank-master > water-tank > tank_animation > index.html > ...

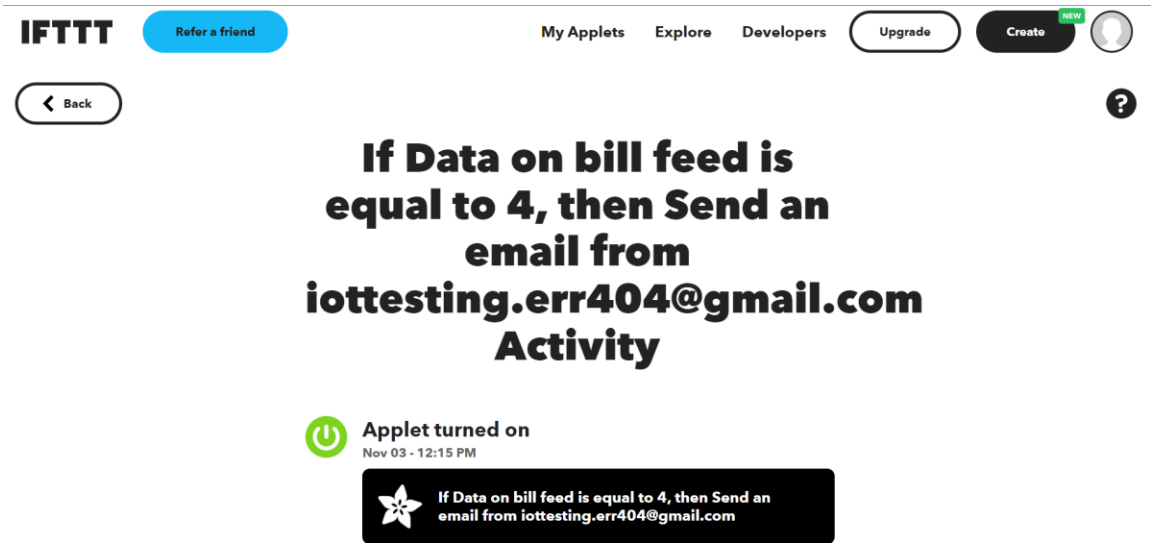
```html
<!doctype html>
<html lang="en">

<head>

<script type="text/javascript" src="https://cdn.fusioncharts.com/fusioncharts/latest/fusioncharts.js"></script>
<script type="text/javascript" src="https://cdn.fusioncharts.com/fusioncharts/latest/themes/fusioncharts.theme.fusion.js"></script>

<script type="text/javascript" src="tank_animation.js"></script>

</head>

<body>

<div style='border:0px solid green;'>

    <div id='chart-container'>Reload Page if you don't see animation</div>

</div>


</body>
</html>
```

D: > Raj Study > 7th Sem > ECE504 > Project > Smart-Water-Tank-master > water-tank > tank_simulation > index.html > html

```html
<html>
<head>
    <title>Simulation</title>
    <script type="text/javascript" src="https://cdn.fusioncharts.com/fusioncharts/latest/fusioncharts.js"></script>
    <script type="text/javascript" src="https://cdn.fusioncharts.com/fusioncharts/latest/themes/fusioncharts.theme.fusion.js"></script>

    <script type="text/javascript" src="tank_animation_simulation.js"></script>
    <script type="text/javascript" src='../basic_page/jquery.min.js'></script>


    <script>

        function data_request_timer(delay){
            window.setInterval(get_data, delay); //timer for initiating ajax request

        }

        function get_data(){
            console.log("fetching data from server");

            $.get("read_data.php?widget=info",
            function(data, status){
                document.getElementById("info").innerHTML = data;
            });

        }

    </script>

</head>
<body onload="data_request_timer(1000)">

<div align='center'>
    <div style='margin-top:30px'>
        <p id='info'></p>
    </div>

    <div style='margin-top:30px'>
        <div id='chart-container'>Reload Page if you don't see animation</div>
    </div>

    <div style='margin-top:30px; width:50%;'>
        <p>In this simulation, the server time seconds info is fed to the tank animation. See how the parameters are tweaked in the tank_animation_simulation.js file to
    </div>
</div>


</body>
</html>
```

Photo of webpage(IFTTT):





Details of Communication protocol:

1. **HTTP:**
   HyperText Transfer Protocol is known as HTTP. It is an access protocol for data on the World Wide Web (www). Data in the form of plain text, hypertext, audio, video, and other formats can all be transferred via the HTTP protocol.
   Because of its effectiveness in a hypertext context where there are quick hops from one page to another, this protocol is also known as the "HyperText Transfer Protocol." Since it also moves files from one site to another, HTTP is comparable to FTP in this regard. However, HTTP is easier to use than FTP since it simply makes one connection and doesn't utilise a control connection to move data. Data in a MIME-like format is transmitted through HTTP. Since data is exchanged between the client and server through HTTP, it is comparable to SMTP. The method that messages are transferred from the client to the server and from the server to the client differs between HTTP and SMTP. While HTTP communications are sent instantly, SMTP messages are saved and forwarded.
2. **TCP:**
   Transmission Control Protocol is referred to as TCP. The packet transmission from source to destination is made easier by this transport layer protocol. Since it is a connection-oriented protocol, communication between computer devices in a network begins with the link being established. Together, they are referred to as a TCP/IP since they are utilised with an IP protocol. The TCP's primary role is to retrieve data from the application layer. The data is then divided into numerous packets, each of which is given a number, before being transmitted to the destination. The packets are reassembled by the TCP and sent to the application layer from the opposite side. Since TCP is a connection-oriented protocol, the connection will continue to exist as long as the sender and receiver are still in contact.
3. **SMTP:**

Simple Mail Transfer Protocol is known as SMTP. The Simple Mail Transfer Protocol (SMTP) is a set of rules for communication that enables applications to send electronic mail over the internet. Based on email addresses, it is an application used to deliver messages to other computer users. Between users using the same or other computers, it offers mail exchange, and it also supports: One message may be sent to one or many recipients. Messages can be sent using text, audio, video, or graphics. The communications may also be sent through networks other than the Internet. Setting up communication rules between servers is the major usage of SMTP.

	Wi-Fi	**Bluetooth**	**Zigbee**
Range	10-100 metres	50-100 metres	10-100 metres
Networking Topology	Ad-hoc, peer to peer, star, or mesh	Point to hub	Ad-hoc, very small networks
Operating Frequency	2.4GHz	2.4 and 5 GHz	2.4GHz
Complexity	Low	High	High
Power Consumption	Very low	High	Medium
Security	128 AES plus application layer security		64 and 128 bit encryption
Typical Applications	Industrial control and monitoring, sensor networks, building automation, home control and automation, toys, games	Wireless LAN connectivity, broadband Internet access	Wireless connectivity between devices such as phones, PDA, laptops, headsets

Complete Program:
For IoT Raspberry Pi Smart Container:

Code to communicate with sensors:

```
def readCount():

    i=0

    Count=0

    gpio.setup(DT, gpio.OUT)

    gpio.output(DT,1)

    gpio.output(SCK,0)

    gpio.setup(DT, gpio.IN)


    while gpio.input(DT) == 1:

        i=0

    for i in range(24):

            gpio.output(SCK,1)

            Count=Count<<1

            gpio.output(SCK,0)


            if gpio.input(DT) == 0:

                Count=Count+1

                #print Count


    gpio.output(SCK,1)

    Count=Count^0x800000

    #time.sleep(0.001)

    gpio.output(SCK,0)

    return Count

```

Code to communicate with actuators and displays:

LED:

```
def begin():

  lcdcmd(0x33)
  lcdcmd(0x32)
  lcdcmd(0x06)
  lcdcmd(0x0C)
  lcdcmd(0x28)
  lcdcmd(0x01)
  time.sleep(0.0005)


def lcdcmd(ch):
  gpio.output(RS, 0)
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)

  if ch&0x10==0x10:
    gpio.output(D4, 1)

  if ch&0x20==0x20:
    gpio.output(D5, 1)

  if ch&0x40==0x40:
    gpio.output(D6, 1)

  if ch&0x80==0x80:
    gpio.output(D7, 1)

  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)

 # Low bits

  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)

  if ch&0x01==0x01:
    gpio.output(D4, 1)

  if ch&0x02==0x02:
    gpio.output(D5, 1)

  if ch&0x04==0x04:
    gpio.output(D6, 1)

  if ch&0x08==0x08:
    gpio.output(D7, 1)

  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)
```

```
60  def lcdwrite(ch):
61
62      gpio.output(RS, 1)
63      gpio.output(D4, 0)
64      gpio.output(D5, 0)
65      gpio.output(D6, 0)
66      gpio.output(D7, 0)
67
68      if ch&0x10==0x10:
69          gpio.output(D4, 1)
70
71      if ch&0x20==0x20:
72          gpio.output(D5, 1)
73
74      if ch&0x40==0x40:
75          gpio.output(D6, 1)
76
77      if ch&0x80==0x80:
78          gpio.output(D7, 1)
79
80      gpio.output(EN, 1)
81      time.sleep(0.005)
82      gpio.output(EN, 0)
83
84
85      # Low bits
86
87      gpio.output(D4, 0)
88      gpio.output(D5, 0)
89      gpio.output(D6, 0)
90      gpio.output(D7, 0)
91
92      if ch&0x01==0x01:
93          gpio.output(D4, 1)
94
95      if ch&0x02==0x02:
96          gpio.output(D5, 1)
97
98      if ch&0x04==0x04:
99          gpio.output(D6, 1)
100
101     if ch&0x08==0x08:
102         gpio.output(D7, 1)
103
104     gpio.output(EN, 1)
105     time.sleep(0.005)
106     gpio.output(EN, 0)
107
108
109 def lcdclear():
110     lcdcmd(0x01)
111
112
```

```
113   def lcdprint(Str):
114
115     l=0;
116     l=len(Str)
117
118     for i in range(l):
119       lcdwrite(ord(Str[i]))
120
121
122
123   def setCursor(x,y):
124
125       if y == 0:
126           n=128+x
127
128       elif y == 1:
129           n=192+x
130
131       lcdcmd(n)
132
```

Send Mail:

```
1  def show_weight():
2
3    count= readCount()
4    w=0
5    w=(count-sample)/106
6    print(w,"g")
7    setCursor(0,0)
8    data=str(w);
9    lcdprint(data)
10   lcdprint("g   ")
11   global flag
12
13   if w>300:
14       if flag == 0:
15         lcdclear()
16         lcdprint("Container Is Full")
17         setCursor(0,1);
18         lcdprint("Sending Email")
19         server = smtplib.SMTP('smtp.mail.yahoo.com', 587)
20         server.starttls()
21         server.login("iottesting.err404@yahoo.com", "password")
22         msg = "Alert.... Container Full"
23         server.sendmail("iottesting.err404@yahoo.com", "shethpurvam@yahoo.com", msg)
24         server.quit()
25         lcdclear()
26         lcdprint("Email Sent")
27         flag=1;
28         lcdclear()
29
30   elif w<30:
31     if flag==1:
32       lcdclear()
33       lcdprint("Container Empty")
34       setCursor(0,1);
35       lcdprint("Sending Email")
36       server = smtplib.SMTP(''smtp.mail.yahoo.com', 587)
37       server.starttls()
38       server.login("iottesting.err404@yahoo.com", "password")
39       msg = "Alert.... Container is Empty"
40       server.sendmail("iottesting.err404@yahoo.com", "shethpurvam@yahoo.com", msg)
41       server.quit()
42       lcdclear()
43       lcdprint("Email Sent")
44       flag=0;
45       lcdclear()
46
47   return jsonify(result=w)
48
```

The size of the code of the “index.html” file is: 917bytes.

The size of the code of the “pi.py” file is: 5.22KB.

The average execution time for the above-mentioned code is approximately 20 seconds.

For Smart Water Tank:

Sensor.py file

```python
import RPi.GPIO as GPIO
import time,os
import datetime

TRIG = 6
ECHO = 5
ALARM = 23

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG, False)

GPIO.setup(ALARM,GPIO.OUT)
GPIO.output(ALARM, True)

print ("Waiting For Sensor To Settle")
time.sleep(1)

def get_distance():
    dist_add = 0
    for x in range(20):
        try:
            GPIO.output(TRIG, True)
            time.sleep(0.00001)
            GPIO.output(TRIG, False)

            while GPIO.input(ECHO)==0:
                pulse_start = time.time()

            while GPIO.input(ECHO)==1:
                pulse_end = time.time()

            pulse_duration = pulse_end - pulse_start

            distance = pulse_duration * 17150

            distance = round(distance, 3)
            print (x, "distance: ", distance)

            dist_add = dist_add + distance

            time.sleep(.1)

        except Exception as e:

            pass

    avg_dist=dist_add/20
    dist=round(avg_dist,3)

    return dist

def sendData_to_remoteServer(dist):
```

```
72
73  def main():
74
75      distance=get_distance()
76
77      print ("distance: ", distance)
78      sendData_to_remoteServer(distance)
79      low_level_warning(distance)
80      print ("---------------------")
81
82  if __name__ == '__main__':
83      main()
```

Setup_cron.sh file

```
1  FILE_NAME="sensor.py"
2
3  crontab -u $USER -l| grep $PWD/$FILE_NAME > /dev/null
4  if [ $? -eq 0 ]
5  then
6      echo "$FILE_NAME is already running"
7  else
8      crontab -l > mycron
9      echo "* * * * * sudo python $PWD/$FILE_NAME &" >> mycron
10     crontab mycron
11     rm mycron
12     echo "cron task added. Now $PWD/$FILE_NAME file will run every minute"
13 fi
```

From sensor.py

```
56  def sendData_to_remoteServer(dist):
57
58      url_remote="http://192.168.172.92/water-tank/insert_data.php?dist=" + str(dist)
59      cmd="curl -s " + url_remote
60      result=os.popen(cmd).read()
61      print (cmd)
62
63  def low_level_warning(dist):
64      tank_height=114
65      level=tank_height-dist
66      if(level<40):
67          print("level low : ", level)
68          GPIO.output(ALARM, False)
69      else:
70          GPIO.output(ALARM, True)
71          print("level ok")
72
73  def main():
74
75      distance=get_distance()
76
77      print ("distance: ", distance)
78      sendData_to_remoteServer(distance)
79      low_level_warning(distance)
80      print ("---------------------")
81
82  if __name__ == '__main__':
83      main()
```

index.html file

```html
<!doctype html>
<html lang="en">

<head>
<script type="text/javascript" src='jquery.min.js'></script>

<script>
    function data_request_timer(){
        window.setInterval(get_data, 2000);
    }

    function get_data(){
        console.log("fetching data from server");

        $.get("read_data1.php",
        function(data, status){
            document.getElementById("info").innerHTML = data;
        });

    }

</script>
</head>

<body onload="data_request_timer()">

<div style='border:0px solid blue; padding:0%;'>
    <p id='info'></p>
</div>

<button>
    <a style='text-decoration:none;font-size:12px' href='../insert_data.php?dist=65' target='_blank'>Insert dummy value</a>
</button>

</body>
</html>
```

insert_data.php file

```php
<?php
date_default_timezone_set("Asia/Kolkata");

include_once 'util.php';

$val=$_GET["dist"];

insert_in_db($val);

function insert_in_db($val){
   echo"<p>Inserting value <b>$val</b> in database</p>";

    $connection=create_db_connection();

    $t=time();
    $date_time= date('Y-m-d H:i:s', $t);

    echo "<p>$date_time</p>";

    $sql="INSERT INTO `level_log`(`level`, `timestamp`, `date_time`)
                    VALUES ('$val','$t', '$date_time')";

    $result = mysqli_query($connection,$sql);

        if ( !$result ) {
            die(mysql_error()."\n".$sql);
            echo" !!!<br> ";
        }

       if ( $result ) {
            echo" <p>successfully inserted >>>> </p> ";
      }

    close_db_connection($connection);

}
?>
```

The size of the code for the “sensor.py” file is: 1.47KB
The size of the code of “setup_cron.sh” file is: 321bytes
The size of the code of the “index.html” file is: 669bytes
The size of the code of the “insert_data.php” file is: 784bytes
The average execution time for the above-mentioned code is approximately 25 seconds.

For Energy Consumption Monitor:

mqtt_esp8266

```
/*************************************************
  Adafruit MQTT Library ESP8266 Example

  Must use ESP8266 Arduino from:
    https://github.com/esp8266/Arduino

  Works great with Adafruit's Huzzah ESP board & Feather
  ----> https://www.adafruit.com/product/2471
  ----> https://www.adafruit.com/products/2821



  Adafruit invests time and resources providing this open source code,
  please support Adafruit and open-source hardware by purchasing
  products from Adafruit!

  Written by Tony DiCola for Adafruit Industries.
  MIT license, all text above must be included in any redistribution
 *************************************************/
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "Raj's iphone"
#define WLAN_PASS       "12345678"
char watt[5];

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
```

mqtt_esp8266

```
#define AIO_USERNAME    "iottesting_err404"
#define AIO_KEY         "aio_Hflh08LPcL6U5NwNQ52IZzrd77Bl"

/************ Global State (you don't need to change this!) ******************/

WiFiClient client;

int bill_amount = 0;

unsigned int energyTariff = 8.0;

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Publish Power = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/Power");

Adafruit_MQTT_Publish bill = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/bill");

void MQTT_connect();


void setup() {

  Serial.begin(115200);

  delay(10);

  Serial.println(F("Adafruit MQTT demo"));


  // Connect to WiFi access point.

  Serial.println(); Serial.println();
```

mqtt_esp8266

```
  Serial.print("Connecting to ");

  Serial.println(WLAN_SSID);


  WiFi.begin(WLAN_SSID, WLAN_PASS);

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println();


  Serial.println("WiFi connected");

  Serial.println("IP address: "); Serial.println(WiFi.localIP());

}


void loop() {

  // Ensure the connection to the MQTT server is alive (this will make the first

  // connection and automatically reconnect when disconnected).  See the MQTT_connect

  // function definition further below.
```

mqtt_esp8266

```
 MQTT_connect();


int i=0;

float watt1;

if(Serial.available() > 0 ){

  delay(100); //allows all serial sent to be received together

   while(Serial.available() && i<5) {

    watt[i++] = Serial.read();

   }

   watt[i++]='\0';

 }


watt1 = atof(watt);

 bill_amount = watt1 * (energyTariff/1000);      // 1unit = 1kwH

 Serial.print(F("\nSending Power val "));

 Serial.println(watt1);

 Serial.print("...");
```

mqtt_esp8266

```
  if (! Power.publish(watt1)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

   if (! bill.publish(bill_amount)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }

if (bill_amount==4){
for (int i =0; i<=2; i++)
{
  bill.publish(bill_amount);
delay(5000);
}
```

mqtt_esp8266

```
bill_amount =6;

}


delay(5000);

}

// Function to connect and reconnect as necessary to the MQTT server.

// Should be called in the loop function and it will take care if connecting.

void MQTT_connect() {

  int8_t ret;


  // Stop if already connected.

  if (mqtt.connected()) {

    return;

  }


  Serial.print("Connecting to MQTT... ");
```

mqtt_esp8266

```
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```

The size of the code of the “sensor.iso” file is: 567bytes
The size of the code of the “mqtt_esp8266.iso” file is: 4.10KB
The average execution time for the above-mentioned code is approximately 30 seconds.

For Room Cleaner Robot:

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
// ---------------------------------------------------------------------------
// Created by Tim Eckel - teckel@leethost.com
// Copyright 2012 License: GNU GPL v3 http://www.gnu.org/licenses/gpl-3.0.html
//
// See "NewPing.h" for purpose, syntax, version history, links, and more.
// ---------------------------------------------------------------------------

#include "NewPing.h"


// ---------------------------------------------------------------------------
// NewPing constructor
// ---------------------------------------------------------------------------

NewPing::NewPing(uint8_t trigger_pin, uint8_t echo_pin, int max_cm_distance) {
    _triggerBit = digitalPinToBitMask(trigger_pin); // Get the port register bitmask for the trigger pin.
    _echoBit = digitalPinToBitMask(echo_pin);       // Get the port register bitmask for the echo pin.

    _triggerOutput = portOutputRegister(digitalPinToPort(trigger_pin)); // Get the output port register for the trigger pin.
    _echoInput = portInputRegister(digitalPinToPort(echo_pin));         // Get the input port register for the echo pin.

    _triggerMode = (uint8_t *) portModeRegister(digitalPinToPort(trigger_pin)); // Get the port mode register for the trigger pin.

    _maxEchoTime = min(max_cm_distance, MAX_SENSOR_DISTANCE) * US_ROUNDTRIP_CM + (US_ROUNDTRIP_CM / 2); // Calculate the maximum distance in uS.

#if DISABLE_ONE_PIN == true
    *_triggerMode |= _triggerBit; // Set trigger pin to output.
#endif
}


// ---------------------------------------------------------------------------
// Standard ping methods
// ---------------------------------------------------------------------------

unsigned int NewPing::ping() {
    if (!ping_trigger()) return NO_ECHO;               // Trigger a ping, if it returns false, return NO_ECHO to the calling function.
    while (*_echoInput & _echoBit)                     // Wait for the ping echo.
        if (micros() > _max_time) return NO_ECHO;      // Stop the loop and return NO_ECHO (false) if we're beyond the set maximum distance.
    return (micros() - (_max_time - _maxEchoTime) - 5); // Calculate ping time, 5uS of overhead.
}


unsigned int NewPing::ping_in() {
    unsigned int echoTime = NewPing::ping();          // Calls the ping method and returns with the ping echo distance in uS.
    return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

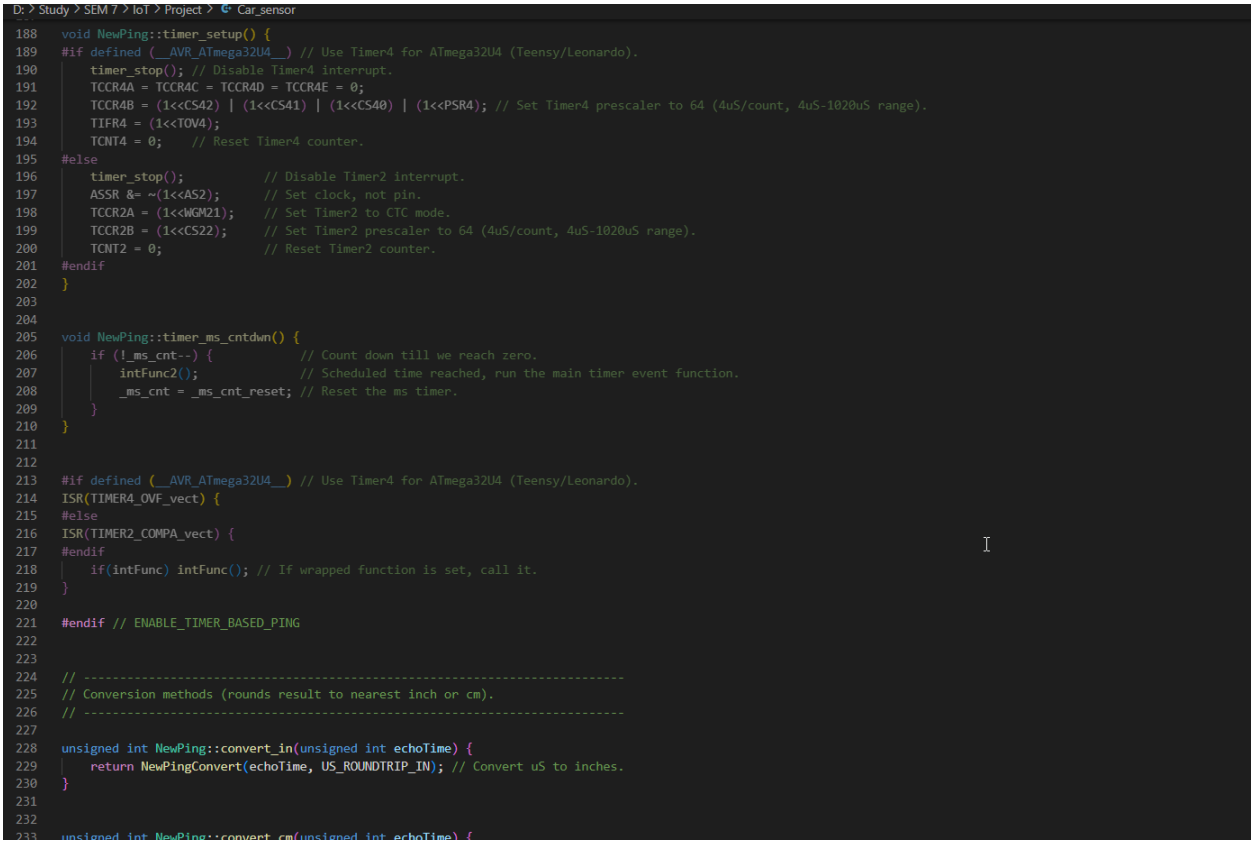
```
    return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
}


unsigned int NewPing::ping_cm() {
    unsigned int echoTime = NewPing::ping();          // Calls the ping method and returns with the ping echo distance in uS.
    return NewPingConvert(echoTime, US_ROUNDTRIP_CM); // Convert uS to centimeters.
}


unsigned int NewPing::ping_median(uint8_t it) {
    unsigned int uS[it], last;
    uint8_t j, i = 0;
    uS[0] = NO_ECHO;
    while (i < it) {
        last = ping();           // Send ping.
        if (last == NO_ECHO) {   // Ping out of range.
            it--;                // Skip, don't include as part of median.
            last = _maxEchoTime; // Adjust "last" variable so delay is correct length.
        } else {                     // Ping in range, include as part of median.
            if (i > 0) {             // Don't start sort till second ping.
                for (j = i; j > 0 && uS[j - 1] < last; j--) // Insertion sort loop.
                    uS[j] = uS[j - 1]; // Shift ping array to correct position for sort insertion.
            } else j = 0;            // First ping is starting point for sort.
            uS[j] = last;            // Add last ping to array in sorted position.
            i++;                     // Move to next ping.
        }
        if (i < it) delay(PING_MEDIAN_DELAY - (last >> 10)); // Millisecond delay between pings.
    }
    return (uS[it >> 1]); // Return the ping distance median.
}


// ---------------------------------------------------------------------------
// Standard ping method support functions (not called directly)
// ---------------------------------------------------------------------------

boolean NewPing::ping_trigger() {
#if DISABLE_ONE_PIN != true
    *_triggerMode |= _triggerBit;    // Set trigger pin to output.
#endif
    *_triggerOutput &= ~_triggerBit; // Set the trigger pin low, should already be low, but this will make sure it is.
    delayMicroseconds(4);            // Wait for pin to go low, testing shows it needs 4uS to work every time.
    *_triggerOutput |= _triggerBit;  // Set trigger pin high, this tells the sensor to send out a ping.
    delayMicroseconds(10);           // Wait long enough for the sensor to realize the trigger pin is high. Sensor specs say to wait 10uS.
    *_triggerOutput &= ~_triggerBit; // Set trigger pin back to low.
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
#if DISABLE_ONE_PIN != true
    *_triggerMode &= ~_triggerBit;   // Set trigger pin to input (when using one Arduino pin this is technically setting the echo pin to input as both are tied to the
#endif

    _max_time =  micros() + MAX_SENSOR_DELAY;                // Set a timeout for the ping to trigger.
    while (*_echoInput & _echoBit && micros() <= _max_time) {} // Wait for echo pin to clear.
    while (!(*_echoInput & _echoBit))                        // Wait for ping to start.
        if (micros() > _max_time) return false;              // Something went wrong, abort.

    _max_time = micros() + _maxEchoTime; // Ping started, set the timeout.
    return true;                         // Ping started successfully.
}


#ifdef ENABLE_TIMER_BASED_PING

// ---------------------------------------------------------------------------
// Timer interrupt ping methods (won't work with ATmega8 and ATmega128)
// ---------------------------------------------------------------------------

void NewPing::ping_timer(void (*userFunc)(void)) {
    if (!ping_trigger()) return;         // Trigger a ping, if it returns false, return without starting the echo timer.
    timer_us(ECHO_TIMER_FREQ, userFunc); // Set ping echo timer check every ECHO_TIMER_FREQ uS.
}


boolean NewPing::check_timer() {
    if (micros() > _max_time) { // Outside the timeout limit.
        timer_stop();           // Disable timer interrupt
        return false;           // Cancel ping timer.
    }

    if (!(*_echoInput & _echoBit)) { // Ping echo received.
        timer_stop();                // Disable timer interrupt
        ping_result = (micros() - (_max_time - _maxEchoTime) - 13); // Calculate ping time, 13uS of overhead.
        return true;                 // Return ping echo true.
    }

    return false; // Return false because there's no ping echo yet.
}


// ---------------------------------------------------------------------------
// Timer2/Timer4 interrupt methods (can be used for non-ultrasonic needs)
// ---------------------------------------------------------------------------
```

D: > Study > SEM 7 > IoT > Project > Car_sensor

```
// Variables used for timer functions
void (*intFunc)();
void (*intFunc2)();
unsigned long _ms_cnt_reset;
volatile unsigned long _ms_cnt;


void NewPing::timer_us(unsigned int frequency, void (*userFunc)(void)) {
    timer_setup();      // Configure the timer interrupt.
    intFunc = userFunc; // User's function to call when there's a timer event.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    OCR4C = min((frequency>>2) - 1, 255); // Every count is 4uS, so divide by 4 (bitwise shift right 2) subtract one, then make sure we don't go over 255 limit.
    TIMSK4 = (1<<TOIE4);                  // Enable Timer4 interrupt.
#else
    OCR2A = min((frequency>>2) - 1, 255); // Every count is 4uS, so divide by 4 (bitwise shift right 2) subtract one, then make sure we don't go over 255 limit.
    TIMSK2 |= (1<<OCIE2A);                // Enable Timer2 interrupt.
#endif
}


void NewPing::timer_ms(unsigned long frequency, void (*userFunc)(void)) {
    timer_setup();                       // Configure the timer interrupt.
    intFunc = NewPing::timer_ms_cntdwn;  // Timer events are sent here once every ms till user's frequency is reached.
    intFunc2 = userFunc;                 // User's function to call when user's frequency is reached.
    _ms_cnt = _ms_cnt_reset = frequency; // Current ms counter and reset value.

#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    OCR4C = 249;         // Every count is 4uS, so 1ms = 250 counts - 1.
    TIMSK4 = (1<<TOIE4); // Enable Timer4 interrupt.
#else
    OCR2A = 249;           // Every count is 4uS, so 1ms = 250 counts - 1.
    TIMSK2 |= (1<<OCIE2A); // Enable Timer2 interrupt.
#endif
}


void NewPing::timer_stop() { // Disable timer interrupt.
#if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
    TIMSK4 = 0;
#else
    TIMSK2 &= ~(1<<OCIE2A);
#endif
}
```



```
D: > Study > SEM 7 > IoT > Project > Car_sensor
188  void NewPing::timer_setup() {
189  #if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
190      timer_stop(); // Disable Timer4 interrupt.
191      TCCR4A = TCCR4C = TCCR4D = TCCR4E = 0;
192      TCCR4B = (1<<CS42) | (1<<CS41) | (1<<CS40) | (1<<PSR4); // Set Timer4 prescaler to 64 (4uS/count, 4uS-1020uS range).
193      TIFR4 = (1<<TOV4);
194      TCNT4 = 0;    // Reset Timer4 counter.
195  #else
196      timer_stop();           // Disable Timer2 interrupt.
197      ASSR &= ~(1<<AS2);      // Set clock, not pin.
198      TCCR2A = (1<<WGM21);    // Set Timer2 to CTC mode.
199      TCCR2B = (1<<CS22);     // Set Timer2 prescaler to 64 (4uS/count, 4uS-1020uS range).
200      TCNT2 = 0;              // Reset Timer2 counter.
201  #endif
202  }
203
204
205  void NewPing::timer_ms_cntdwn() {
206      if (!_ms_cnt--) {           // Count down till we reach zero.
207          intFunc2();             // Scheduled time reached, run the main timer event function.
208          _ms_cnt = _ms_cnt_reset; // Reset the ms timer.
209      }
210  }
211
212
213  #if defined (__AVR_ATmega32U4__) // Use Timer4 for ATmega32U4 (Teensy/Leonardo).
214  ISR(TIMER4_OVF_vect) {
215  #else
216  ISR(TIMER2_COMPA_vect) {
217  #endif
218      if(intFunc) intFunc(); // If wrapped function is set, call it.
219  }
220
221  #endif // ENABLE_TIMER_BASED_PING
222
223
224  // ---------------------------------------------------------------------------
225  // Conversion methods (rounds result to nearest inch or cm).
226  // ---------------------------------------------------------------------------
227
228  unsigned int NewPing::convert_in(unsigned int echoTime) {
229      return NewPingConvert(echoTime, US_ROUNDTRIP_IN); // Convert uS to inches.
230  }
231
232
233  unsigned int NewPing::convert_cm(unsigned int echoTime) {
```

The size of the code of the “NewPing.ino” file is: 11KB
The size of the code of the “AFMotor.ino” file is: 19KB
The size of the code of the “ARDUINO_OBSTACLE_AVOIDING_CAR.ino” file is: 4KB
The average execution time for the above-mentioned code is approximately 30 seconds.

Summary:

Many IoT-based home equipment is included in this project, which might enable the homemaker to create a smart home. The features provided by this project play a crucial role in assisting them since, as is frequently seen in our fast-paced world, nearly every household member is a member of the working class. Additionally, it has been observed that people occasionally fail to properly handle some delicate items, which is unexpected given how this project handles them. The goal of this project is to make homes smarter and to assist by monitoring routine activities and alerting people about the unexpected process. Our concept might be useful for those who are single at home and for homemakers as it helps to prevent terrifying situations by sending them notifications in advance to their mobile phones through email or SMS. Not least among its benefits is that this program aids homes in lowering their carbon footprints.

The future improvement could include following things:

- **UI Enhancement:**
  As the currently made device is in it's early stage the user interface used here is very basic which could be improved in future to make it available to the potential consumers, as well as making it attractive to them.
- **Cost efficiency:**
  If the items required to build the project is bought in bulk then there might be a possibility that the effective cost per product would be lower than to buy items for making a single unit.
- **Improve accuracy:**
  Currently, the devices made shows results which are not up to norms that are expected by the potential consumers, which requires some time to have an improvement in future and could play as a leading product for households.
- **Data security:**
  Nowadays, data security has become very much important for everyone in the market. So, this aspect cannot be ignored, and by implementing highly secured protocols into the device the problem can be solved.

References:

1) M. Vijayalakshmi, Smart Vacuum Cleaner. ResearchGate. https://www.researchgate.net/publication/341162912_Smart_Vacuum_Robot
2) Manasa M. Smart Vacuum Cleaner. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S2666285X21000790?via%3Dihub
3) Automatic Vacuum Cleaner Robot Project. NevonProject. https://nevonprojects.com/automatic-vacuum-cleaner-robot-project/
4) A. Lakkanagavi. Smart Kitchen Containers as a Part of Smart Home Appliances Using IOT and Android. MATJOURNALS. https://zenodo.org/record/3259500/files/%286-10%29%20Smart%20Kitchen%20Containers.pdf?download=1
5) A. Jahagirdar. SMART CONTAINER FOR FOOD WASTAGE PREVENTION USING IOT. IRJMETS. https://www.irjmets.com/uploadedfiles/paper/volume2/issue_9_september_2020/3814/1628083151.pdf
6) PRASANNA LAKSHMI. Smart Water Tank: an IoT based Android Application. irejournals. https://irejournals.com/formatedpaper/1700354.pdf
7) IoT Based Electricity Energy Meter using ESP32 & Blynk. how2electronics. https://how2electronics.com/iot-based-electricity-energy-meter-using-esp32-blynk/
8) Naziya Sulthana, Rashmi N, Prakyathi N Y, Bhavana S, K B Shiva Kumar, 2020, Smart Energy Meter and Monitoring System using IoT, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCETESFT – 2020 (Volume 8 – Issue 14), https://www.ijert.org/smart-energy-meter-and-monitoring-system-using-iot

9) Asad Javed. Container-based IoT Sensor Node on Raspberry Pi and the Kubernetes Cluster Framework. Core. https://core.ac.uk/download/pdf/80720439.pdf
10) Suraksha Sanghvi. Smart water tank. slideshare. https://www.slideshare.net/SurakshaSanghavi/smart-water-tank

Appendix A:

Datasheet for HX711 Load Cell Amplifier: https://pdf1.alldatasheet.com/datasheet-pdf/view/1371088/SPARKFUN/HX711.html

Datasheet for ACS712 Current Sensor: https://pdf1.alldatasheet.com/datasheet-pdf/view/168326/ALLEGRO/ACS712.html

Datasheet for HC-SR04 Sensor: https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf

Datasheet for ESP8266 WiFi Module: https://pdf1.alldatasheet.com/datasheet-pdf/view/1148031/ESPRESSIF/ESP8266.html

Appendix B:

Python vs Java:

- Unlike Java, which will display a syntax error if curly brackets or a semicolon are not added to the program, Python does not need these additions.
- Python is dynamically typed, therefore all one needs to do to determine the data type of a variable at runtime is to assign a value; in contrast to Java, where the data type must be specified explicitly.
- In contrast to Java, which is entirely built on object-oriented and class-based programming models, Python supports a variety of programming models, including imperative, object-oriented, and procedural programming.
- In contrast to Java, which has a high learning curve because of its preset complicated syntaxes, Python is simple to read and learn, which benefits novices who want to grasp the basics of programming rapidly.
- In contrast to Java, which may need a paid license to be utilized for large-scale application development, Python is free and open-source, meaning that its code is available to the public on repositories and that it is free for commercial applications.

Python vs C++:

- Python's automated garbage collection makes it more memory-efficient than C++, which does not have garbage collection.
- Compared to C++, whose sophisticated syntax makes it difficult to grasp and use, Python code is simple to learn, use, and write.
- Python runs easily on practically every computer and operating system because it utilizes an interpreter to run the code. as opposed to C++ code, which does not execute until it has been compiled on the target machine.
- In contrast to C++, which is hard to utilize for quick application development due to its big code fragments, Python may be used for rapid application development simply due to its reduced code size.
- In contrast to C++, where the scope of variables is restricted to the loop, variables declared in Python are freely accessible outside of the loop.

C vs Java:

	C	Java
Programming Model	C is a procedural programming language	Java is an object-oriented programming language.
Platform Dependence	C is platform-dependent.	Java is platform-independent.
Type of language	As it connects the gaps between machine-level and high-level languages, C is a middle-level language.	Due to the fact that Java code is converted into machine language via a compiler or interpreter, Java is a high-level language.
Compilation and Interpretation	C is not understood; it is just compiled.	Both Java is compiled and Java is interpreted.
Pointers	C has support for pointers.	Java does not support pointers.
Threading	C is not intrinsically a multithreaded language.	Java supports threading.
Garbage collection	In C, Garbage Collection needs to be done manually.	In Java, Garbage Collector automatically does the Garbage Collection.
Memory Allocation	To do memory allocation in C functions like malloc(), calloc(), etc. can be used. But there is no 'new' keyword in C.	To do memory allocation in Java, the 'new' keyword can be used.
Application	System and application programming are both done in the C programming language.	Java may only be used for application development; system programming is not permitted.
Functional Units	In C, mostly the functional units are functions as it is a procedural programming language.	In Java, the functional units are mostly objects as it is an object-oriented programming language.
Keywords	There are about thirty-two keywords in C.	There are about fifty keywords in Java.
Security	Compared to Java, C is a less secure programming language.	Comparatively speaking, Java is safer than C.
Exception Handling	C does not provide the features of Exception Handling.	The functionalities of Exception Handling are not available in C. Java has capabilities for handling exceptions using the keywords "try," "catch," "final," and so on.
Portability	C is not portable.	Java is a portable programming language.

Robustness	C is not a robust programming language.	Java is a robust programming language.

C vs C++:

OOPS	OOPS ideas like polymorphism, encapsulation, and inheritance are not supported by C.	Polymorphism, encapsulation, and inheritance are all supported OOPS principles in C++, which is an object-oriented programming language.
Set	C is a subset of C++.	C++ is a superset of C. All code of C can run in C++ but vice versa may or may not be true.
keywords	C has 32 keywords.	C++ has 52 keywords.
Procedural vs OOPS	C is a procedural programming language.	C++ supports both procedural as well as object-oriented programming.
Data and Function	Data and Functions are separate in C.	GData and Functions are encapsulated together as objects in C++.
Information Hiding	Information Hiding is not supported in C.	Information Hiding is supported in C++ via encapsulation.
Overloading	Function and Operator overloading is not supported in C.	Function and Operator overloading is supported in C++.
Function	C is a function-driven language.	C++ is Object driven language.
Structure	C structure does not support defining functions.	C++ Structure supports defining functions.
Reference Variables	Reference Variables are not supported in C.	Reference Variables are supported in C++.
Virtual and friend functions	Virtual and friend functions are not supported in C.	Virtual and friend functions are supported in C++.
Exception Handling	Exception handling is not supported in C.	Exception handling is supported in C++.

Appendix C:

Few troubleshooting and debugging faced during the project are as follows:

- We were not able to upload code from Arduino IDE to ESP8266, as well as the port for ESP8266 module was not being discovered. Later on the problem was identified that *CH340* driver was not installed. After discovering the issue *CH340* was installed and then the hardware started working properly.
- In "Smart water tank" we were not able to store data in the SQL database as the schema was different than the data received from Raspberry Pi, due to this we were also having problem in sending data to front-end. Upon discovering the problem, SQL schema was changed accordingly and the storing of data received from Raspberry Pi as well as sending data to front-end run successfully without any error.
- There was a motor connected with the stand of *HC-SR04*, which was supposed to let the sensor rotate on sideways if the front side is blocked by an obstacle. However, on implementing different codes the motor was not working as expected, which we came to know that the motor had aged and when we replaced it with our colleagues' motor it started working properly.

Appendix D:

For IoT Raspberry Pi Smart Container with Email Alert and Web Monitoring:

This Smart Container can tell you about its full or empty status by sending a mail to your Email ID. It can also monitor the weight of the container in real-time using a web browser, which makes it an IoT project where one can monitor the container from anywhere using the internet. This can be done by fitting it on a stand where the house members usually keep their weight sensitive container.

A plastic coating can be applied on the device which works as a rain protection for the device, it will also protect from weather and environmental changes like rain, thunderstorm, etc.

For Smart Water Tank:

This device will help the user to monitor the level of his/her water tank. Any person can easily track the water level of the tank. If it decreases beyond a certain level, the system will send a notification with an auto-generated message to the person to refill the tank. This device can be fit on the upper inner side of the water tank.

A plastic coating can be applied on the device which works as a rain protection for the device. Also, the device will remain inside the water tank there will be no need to protect from external environmental factors.

For Energy Consumption Monitor:

This function is helpful since the product makes it possible to monitor power usage at any time and from any location in the globe. We could show how we can regulate appliances and conserve energy with only one click by integrating the capability of email sending.

Appendix E:

The basic challenges and it's solution that will be there if the project is implemented in real life is as follows:

- The most common challenge that any device will face is the maintenance. It is required to maintain electronic devices with proper care and attention, as these devices are very prone to get damaged if not maintained properly. To overcome this problem, we can provide a regular service to the consumers of the products. Firstly, for few months this service could be provided for free, after then it could cost some charges to the consumer.
- As the name IoT (Internet of Things) contains the word internet, which means all the vulnerability present on the internet can adversely affect the device, and in severe conditions it may even completely damage the device. These problems can be solved by using secured protocols which sends encrypted data with the help of encryption methods like: RSA, AES, such that even the data is breached no information can be leaked.
- Many of the privacy difficulties posed by the IoT go beyond the present limitations on data protection. A large part of this results from the incorporation of technology into our surroundings without our intentional use. This problem can be solved by giving access to data to some privileged people only, as well as trying to maintain the confidentiality of the data.

----------*Thank You*----------