Embedded System and Design

BCT3002

SLOT- D1+TD1

L51+L52

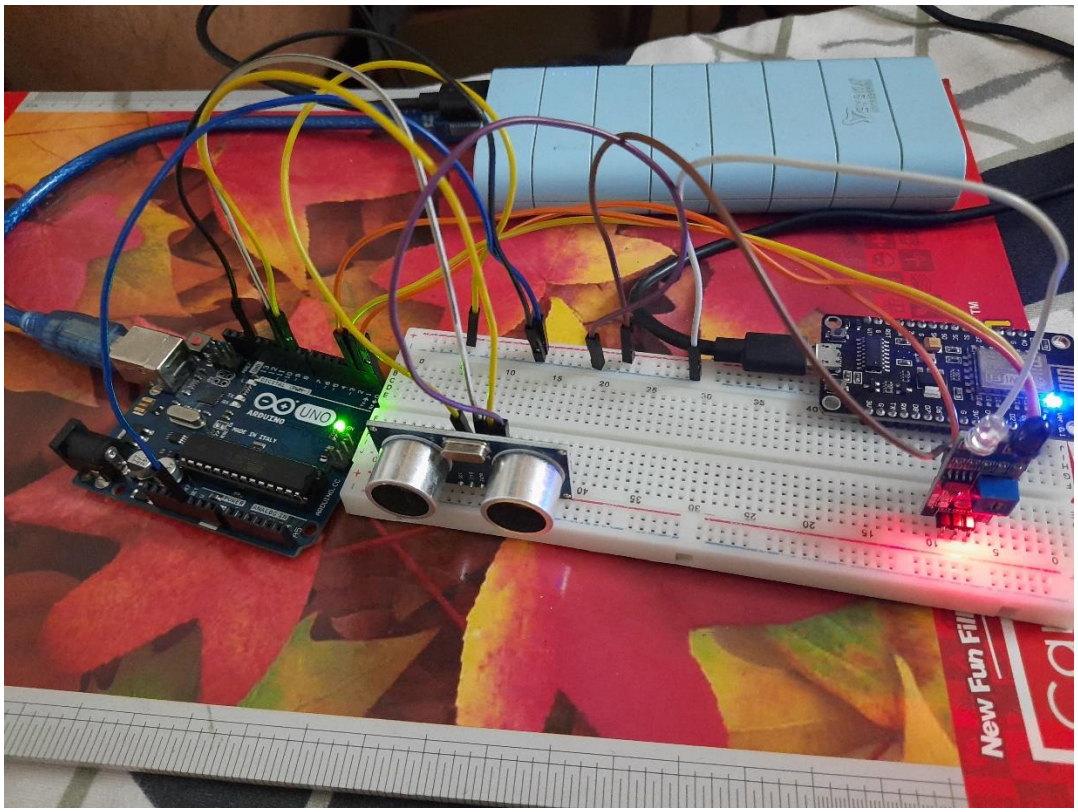
Winter Semester 2021-22

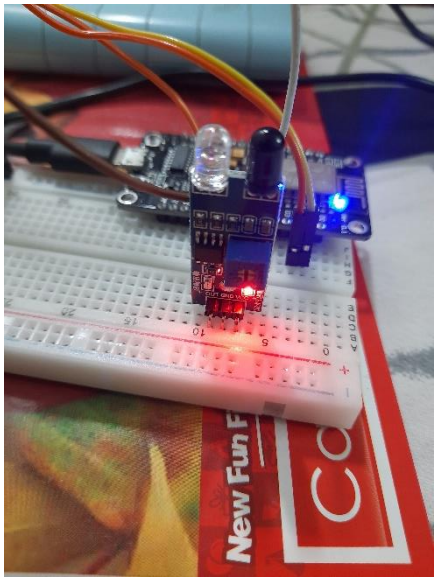
Team Members

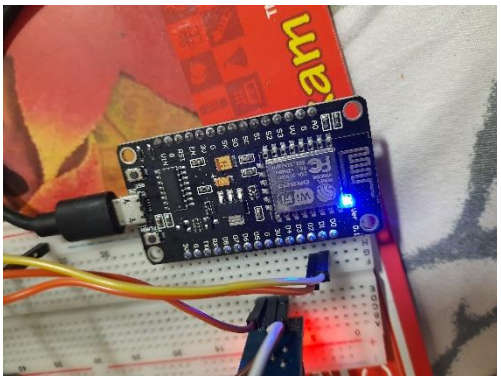
Abhay Sharma (19BCT0125) **Jasshu Garg (19BCT0082)**

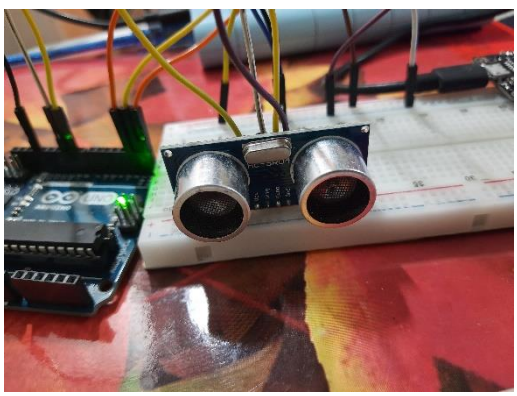
Project Documentation

Hardware Snapshots:









CODE:

```
#include <SPI.h>


int triggerPin = 10;
int echoPin = 9;
int count=0;
int IRSensor = 2;
int capacity=40;
// the setup routine runs once when you press reset:
void setup() {
count=0;
capacity=40;
  Serial.begin(9600);
  pinMode(6, OUTPUT);
pinMode (IRSensor, INPUT);
SPI.begin();


}
// the loop routine runs over and over again forever:
void loop() {
if(count<40)
{


   long sensorReading = readUltrasonicDistance();
  long millimeters = sensorReading/2 * .343;
  if(millimeters <100){
    {
      count++;
      Serial.print("total Cars inside :");
      Serial.println(count);
```

```
      Serial.print("Available spaces are :");
   Serial.println( capacity - count);Serial.println("");
   }
 }
}

if(count>0)
{
 int statusSensor = digitalRead (IRSensor);

 if (statusSensor == 0){

//   if(count>=0 && count<40)
   {count--;
    Serial.print("total Cars inside :");
    Serial.println(count);

      Serial.print("Available spaces are :");
   Serial.println( capacity - count);Serial.println("");
   }

 }

}
 if(count==40)
 {

   Serial.println("Parking Lot is FULL!!!")
 }
delay(200);
}
```

```
long readUltrasonicDistance()
{

  // Clear the trigger
  pinMode(triggerPin, OUTPUT);

  //Send a 10 microsecond "HIGH" signal to prompt the sensor to start
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);


  // Read the echo pin, and return the sound wave travel time in microseconds
  pinMode(echoPin, INPUT);
  long pulseTime = pulseIn(echoPin, HIGH);
  return pulseTime;
}
```

Code for sending real-time data to cloud:

```
Round#include "FS.h"
#include "ESP8266WiFi.h "
#include "PubSubClient.h "
#include "NTPClient.h "
#include "WiFiUdp.h "
#include <string.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(D1,D2);

const char *ssid = "Abhay's Galaxy M30s";
const char *password = "ABHAY2001";

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org");

const char *AWS_endpoint = "a3apchnx1hpbj0-ats.iot.us-west-2.amazonaws.com"; //MQTT broker
ip

void callback(char *topic, byte *payload, unsigned int length)
{
   Serial.print("Message arrived [");
   Serial.print(topic);
   Serial.print("] ");
   for (int i = 0; i< length; i++)
   {
      Serial.print((char)payload[i]);
   }
   Serial.println();
}
WiFiClientSecure espClient;
```

```
PubSubClient client(AWS_endpoint, 8883, callback, espClient); //set MQTT port number to 8883 as
per //standard

long lastMsg = 0;

char msg[50];

int value = 0;


void setup_wifi()

{


  delay(10);

  // We start by connecting to a WiFi network

  espClient.setBufferSizes(512, 512);

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);


  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED)

  {

    delay(500);

    Serial.print(".");

  }


  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());


  timeClient.begin();

  while (!timeClient.update())
```

```
  {
    timeClient.forceUpdate();
  }

  espClient.setX509Time(timeClient.getEpochTime());
}

void reconnect()
{
  // Loop until we're reconnected
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESPthing"))
    {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("ESP8266_Smart_door_log", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");

      char buf[256];
      espClient.getLastSSLError(buf, 256);
      Serial.print("WiFiClientSecure SSL error: ");
```

```
            Serial.println(buf);

            // Wait 5 seconds before retrying
            delay(5000);
          }
        }
      }

      void setup()
      {

        Serial.begin(115200);
        mySerial.begin(9600);
        Serial.setDebugOutput(true);
        // initialize digital pin LED_BUILTIN as an output.
        pinMode(LED_BUILTIN, OUTPUT);
        setup_wifi();
        delay(1000);
        if (!SPIFFS.begin())
        {
          Serial.println("Failed to mount file system");
          return;
        }

        Serial.print("Heap: ");
        Serial.println(ESP.getFreeHeap());

        // Load certificate file
        File cert = SPIFFS.open("/cert.der", "r"); //replace cert.crt eith your uploaded file name
        if (!cert)
        {
```

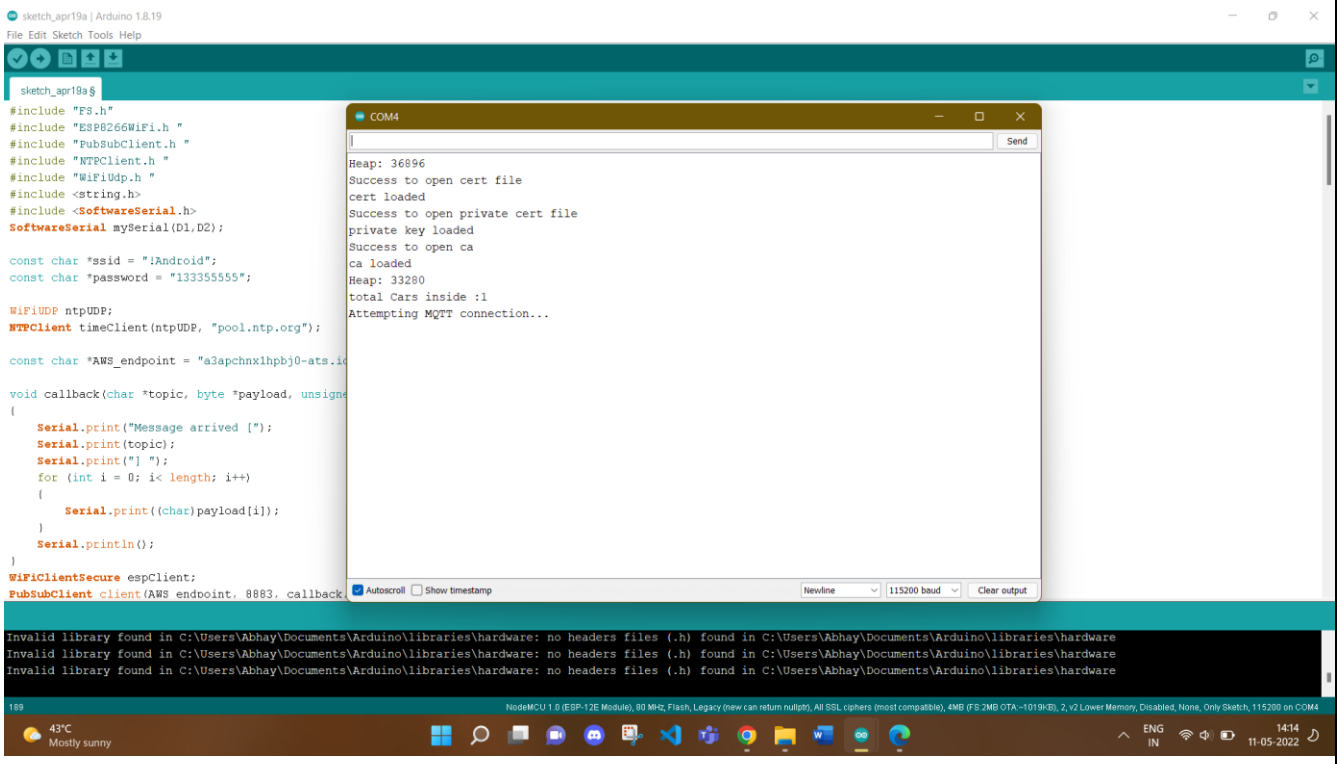
```
  Serial.println("Failed to open cert file");
}
else
  Serial.println("Success to open cert file");

delay(1000);
if (espClient.loadCertificate(cert))
  Serial.println("cert loaded");
else
  Serial.println("cert not loaded");

// Load private key file
File private_key = SPIFFS.open("/private.der", "r"); //replace private eith your uploaded file name
if (!private_key)
{
  Serial.println("Failed to open private cert file");
}
else
  Serial.println("Success to open private cert file");

delay(1000);

if (espClient.loadPrivateKey(private_key))
  Serial.println("private key loaded");
else
  Serial.println("private key not loaded");

// Load CA file
File ca = SPIFFS.open("/ca.der", "r"); //replace ca eith your uploaded file name
if (!ca)
{
```

```
    Serial.println("Failed to open ca ");
  }
  else
    Serial.println("Success to open ca");

  delay(1000);

  if (espClient.loadCACert(ca))
    Serial.println("ca loaded");
  else
    Serial.println("ca failed");

  Serial.print("Heap: ");
  Serial.println(ESP.getFreeHeap());
}

void loop()
{

 String msg1 = mySerial.readStringUntil('\r');
  Serial.println(msg1);
  if (!client.connected()) {
    reconnect();
   }
client.loop();



Serial.print("Publish message: ");

snprintf (msg, 75, "{\"message\": \"%s\"}", msg1.c_str());
Serial.println(msg1);
```

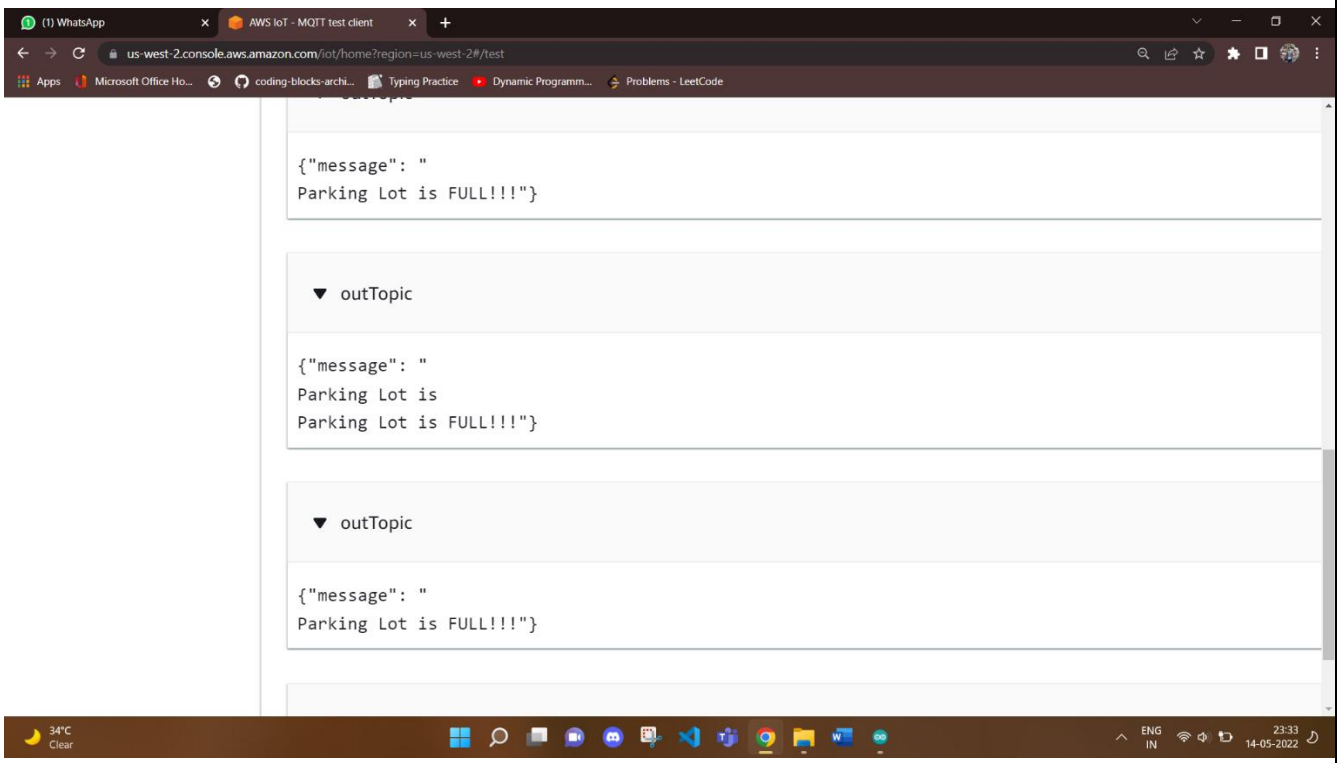
```
//if(msg1!="" or  sizeof(msg1)!=1 )
if(strlen(msg1.c_str())/sizeof('a')>2 )
{
  client.publish("outTopic", msg);
}

Serial.print("Heap: ");
Serial.println(ESP.getFreeHeap()); //Low heap can cause problems                // wait for a second

digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
delay(100); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
delay(100); // wait for a second
}
```

Results:

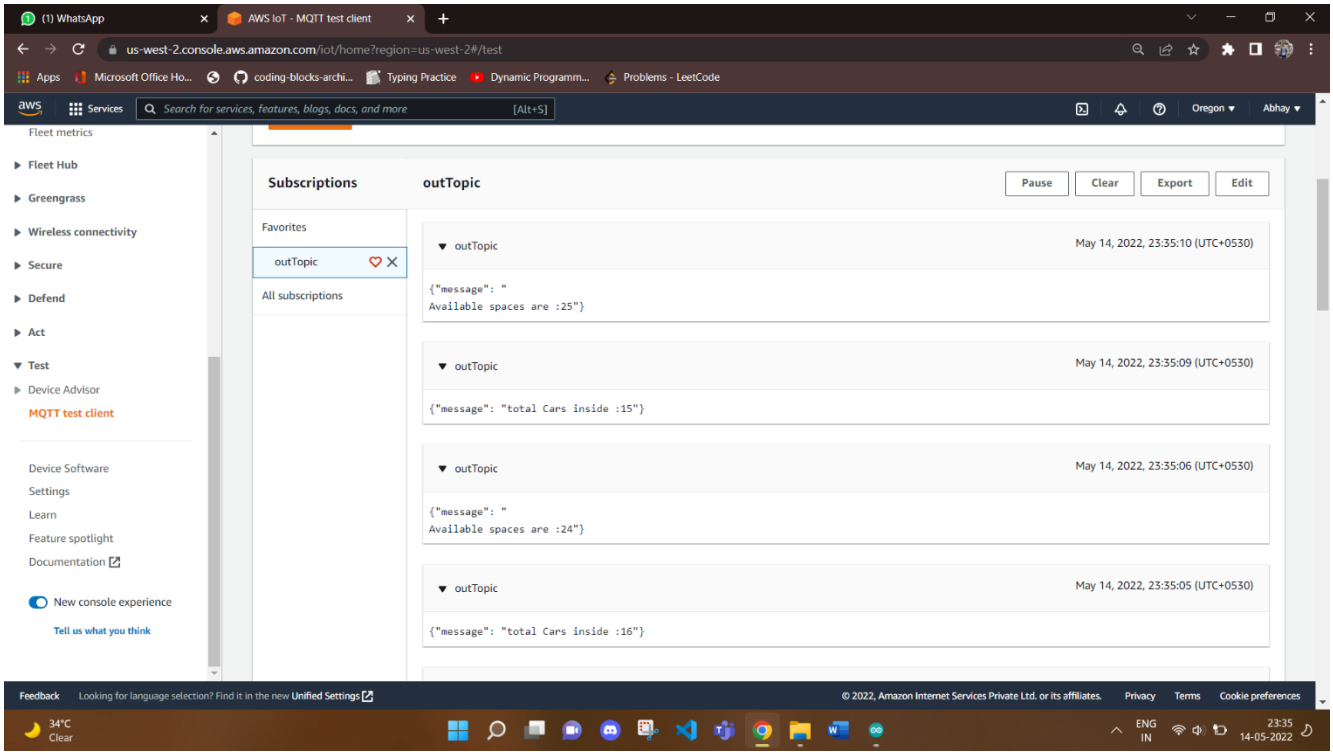
Connecting to cloud:



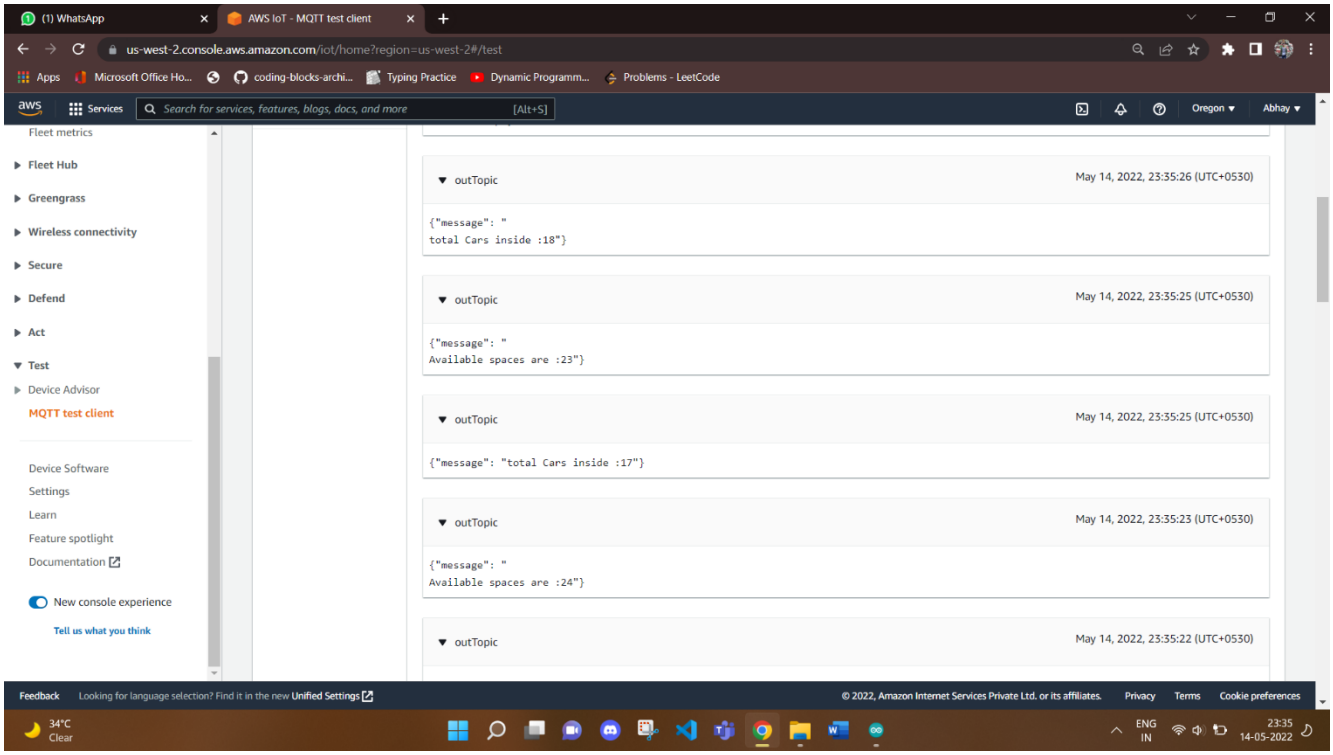
Parking lot full:



Car exiting:

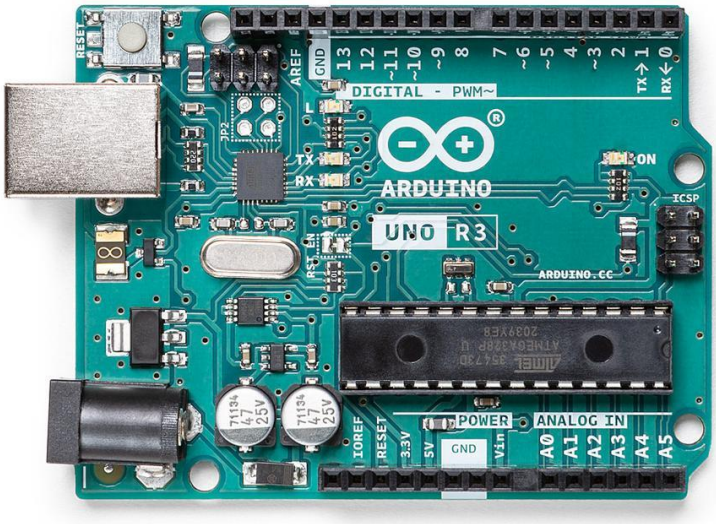


Car entering:



COMPONENTS:

Arduino Uno Board



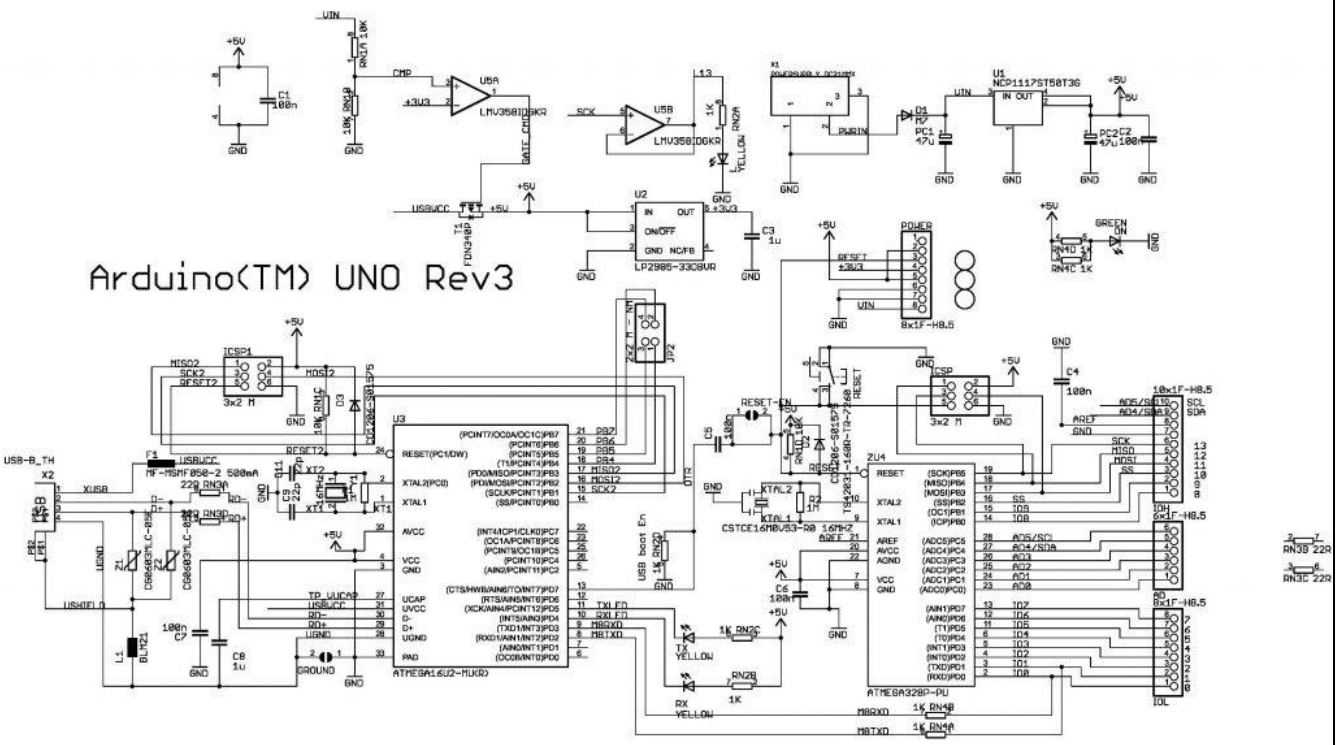
Arduino Uno is a microcontroller board based on the ATmega328P

MICROCONTROLLER	ATmega328P
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	14 (of which 6 provide PWM output)
PWM DIGITAL I/O PINS	6
ANALOG INPUT PINS	6
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
CLOCK SPEED	16 MHz
LED_BUILTIN	13

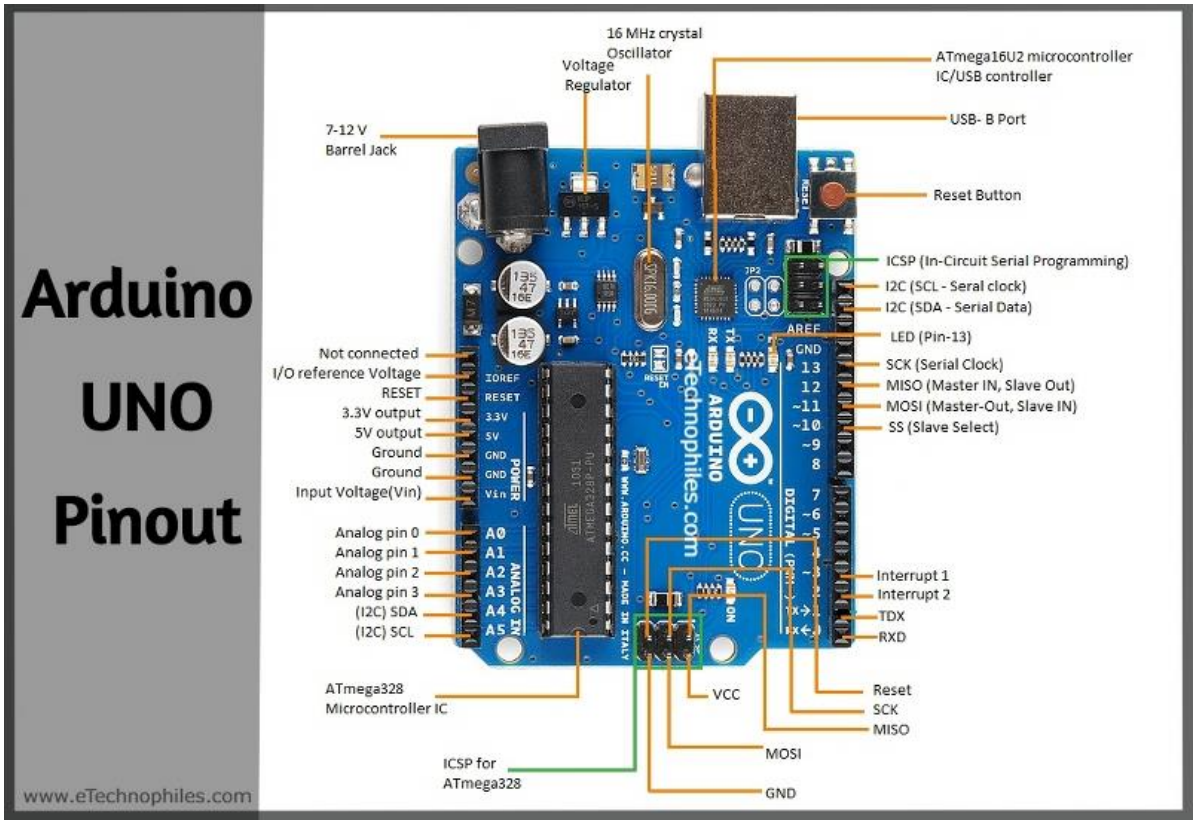
Processing Element: ATmega328P

Max ADC Resolution (bits)	**10**
Program Memory Size (KB)	**32**
Capture/Compare/PWM (CCP)	**1**
Number of Comparators	**1**
CPU Speed (MIPS/DMIPS)	**20**
Data EEPROM (bytes)	**1024**
DigitalTimerQty_16bit	**1**
Max 8 Bit Digital Timers	**2**
Ethernet	**None**
I2C	**1**
Program Memory Type	**Flash**
ADC Channels	**8**
Low Power	**Yes**
Operating Voltage	**1.8 - 5.5**
Output comparator PWM	**6**
Pin Count	**32**

Schematic Diagram



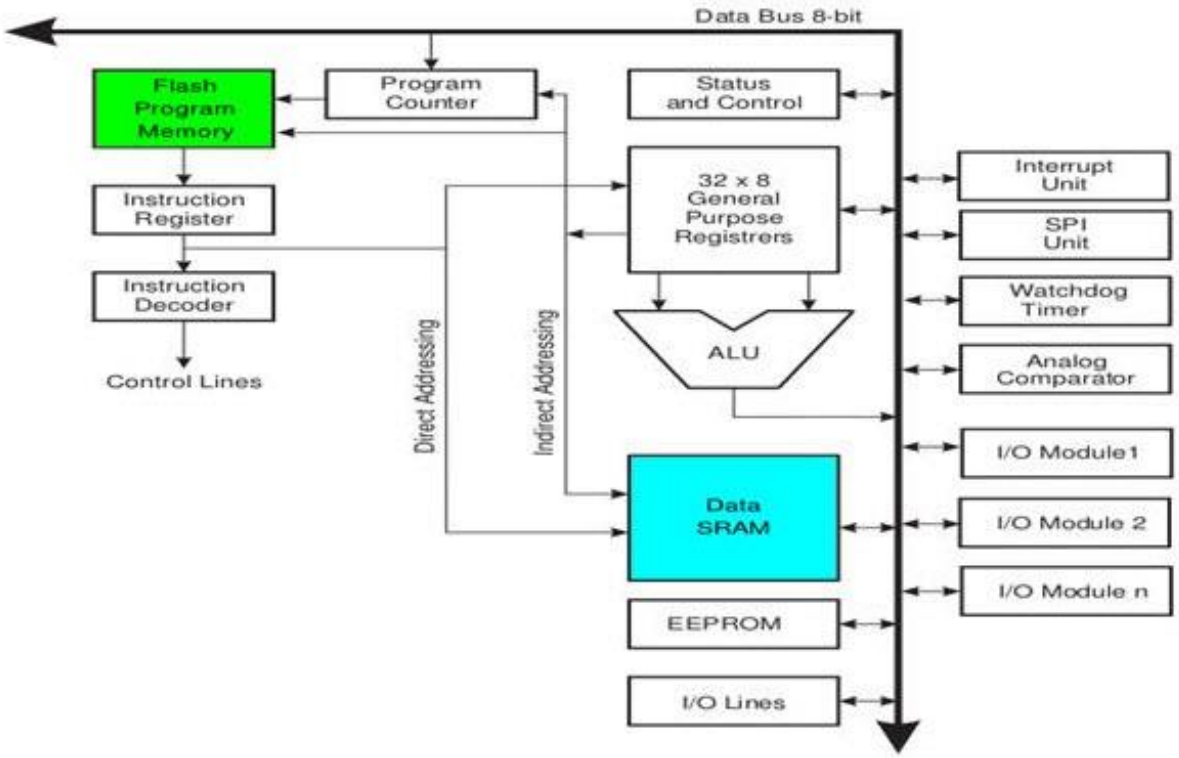
Pin Diagram



Pin details

IOREF (only in R3)	**Input Output voltage reference pin, it is internally connected to 5V. Arduino Shields can read this pin to see the board is running at 3.3V or 5V. This allows it to select proper power source or enable input output voltage translators for proper working.**
RESET	**This pin can be used to RESET arduino. LOW voltage level at this pin resets arduino which is similar to pressing RESET button.**
3.3V	**3.3V Output**
5V	**5V Output**
GND	**Reference Ground**
Vin	**External Power Input. You may connect +ive of the battery to this pin and -ive of the battery to GND.**
A0 – A5	**Analog Voltage Input Pins**
AREF	**Analog Voltage Reference Input. By default analog to digital converter reference voltages are 0 and 5V. We can change the higher reference voltage 5V using AREF pin and analogReference() function.**
0 – 13	**Digital Input Output Pins**
~	**Indicated PWM Outputs**
TX & RX	**Transmit and Receive pins of UART.**

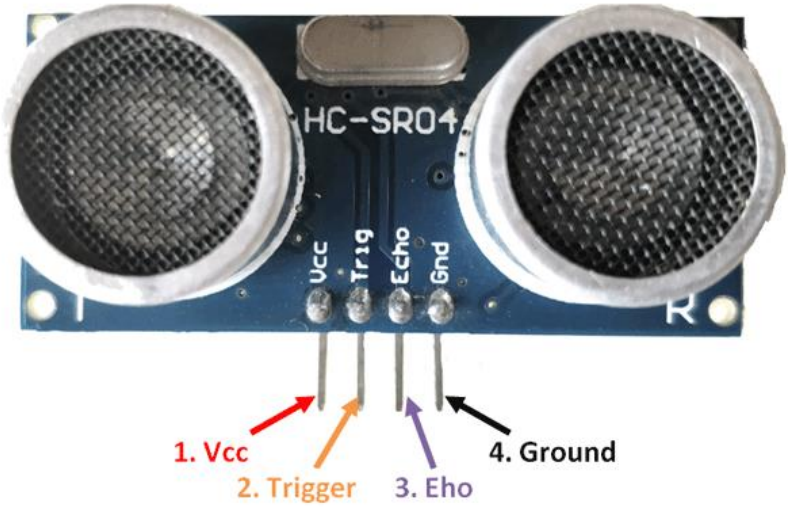
Arduino Architecture



Memory

The RAM in Arduino is like any other RAM, used to store temporary data and is also Volatile. Flash and EEPROM are two types of ROM Memory used to store application code and small data. They are non-volatile in nature.

HC-SR04 Ultrasonic Sensor



Ultrasonic Sensor Pinout Configuration

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

HC-SR04 Sensor Features

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

HC-SR04 Ultrasonic Sensor - Working

As shown above the **HC-SR04 Ultrasonic (US) sensor** is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module