Smart parking

Requirements

- Raspberry Pi
- Arduino Uno
- Jumper
- Ultra Sonic sensor -3
- USB cable
- ESP8266 Wi-Fi module

Raspberry Pi:

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse.

It is used as hardware for various devices like iot which act as minicomputer.

For building the model for my project i use raspberry Pi as hardware device and this helps to give better support for my project as hardware.

Ultra Sonic sensor:

For better and efficient data collection about parking slot we use ultrasonic sensor in our project.

IR sensor can be used but it do not works as efficient in sun light as ultra Sonic sensor.

Ultra Sonic sensor gives as best result about empty spaces to park .so we use this sensor to identify the space for parking.

For small implementation of our project we use only 3 ultra Sonic sensor.

Arduino Uno:

Arduino Uno is a microcontroller which is act as a brain of the project. That having pins that are used to connect devices to it.

For our project we bonds the raspberry Pi to arduino Uno by using bread board and connect with careful things like pin numbers,ground, VCC etc.

Following are the steps for a smart parking system using Raspberry Pi and an ultrasonic sensor:

1. Connect the VCC pin of the ultrasonic sensor to the 5V pin of the Raspberry Pi.
2. Connect the GND pin of the ultrasonic sensor to the GND pin of the Raspberry Pi.
3. Connect the TRIG pin of the ultrasonic sensor to GPIO pin 17 (BCM) of the Raspberry Pi.
4. Connect the ECHO pin of the ultrasonic sensor to GPIO pin 27 (BCM) of the Raspberry Pi.

Above steps can help as to connect the ultra Sonic sensor and raspberry Pi.

Code

```
 #include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11);
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
Const int trig_1 = 2;
Const int echo_1 = 3;
Const int trig_2 = 4;
Const int echo_2 = 5;
Const int trig_3 = 6;
Const int echo_3 = 7;
Float distanceCM_1 = 0, resultCM_1 = 0;
Float distanceCM_2 = 0, resultCM_2 = 0;
Float distanceCM_3 = 0, resultCM_3 = 0;
Long Time_1, Time_2, Time_3;
Float car_1, car_2, car_3;
Float Dist_1 = 8.0, Dist_2 = 8.0, Dist_3 = 8.0;
Int total = 0, timer_cnt = 0;
Void setup()
{
  mySerial.begin(115200);
  pinMode(trig_1, OUTPUT);
  pinMode(trig_2, OUTPUT);
  pinMode(trig_3, OUTPUT);
  pinMode(echo_1, INPUT);
  pinMode(echo_2, INPUT);
  pinMode(echo_3, INPUT);
  digitalWrite(trig_1, LOW);
  digitalWrite(trig_2, LOW);
  digitalWrite(trig_3, LOW);
  lcd.init();
  lcd.backlight();
```

```
  lcd.setCursor(0, 0);
  lcd.print("  IoT CAR PARK");
  lcd.setCursor(0, 1);
  lcd.print(" MONITOR SYSTEM");
  delay(2000);
  lcd.clear();
}

Void loop()
{
  Total = 0;
  Car_1 = sensor_1();
  Car_2 = sensor_2();
  Car_3 = sensor_3();
  Lcd.setCursor(0, 0);
  Lcd.print("CAR1:");
  If (car_1 <= Dist_1)
  {
    Lcd.print("OK ");
  }
  Else
  {
    Total += 1;
  }
  If (car_1 > Dist_1)   lcd.print("NO ");
  Lcd.print("CAR2:");
  If (car_2 <= Dist_2)
  {
```

```
  Lcd.print(“OK “);
}
Else
{
  Total += 1;
}
If (car_2 > Dist_2)   lcd.print(“NO “);
Lcd.setCursor(0, 1);
Lcd.print(“CAR3:”);
If (car_3 <= Dist_3)
{
  Lcd.print(“OK “);
}
Else
{
  Total += 1;
}
If (car_3 > Dist_3)   lcd.print(“NO “);
Lcd.print(“FREE:”);
Lcd.print(total);
If (timer_cnt >= 50)
{
  mySerial.print(‘*’);
  mySerial.print(total);
  mySerial  .println(‘#’);
  timer_cnt = 0;
}
Timer_cnt += 1;
```

```
  Delay(200);
}

Float sensor_1(void)
{
  digitalWrite(trig_1, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_1, LOW);
  Time_1 = pulseIn(echo_1, HIGH);
  distanceCM_1 = Time_1 * 0.034;
  return resultCM_1 = distanceCM_1 / 2;
}

Float sensor_2(void)
{
  digitalWrite(trig_2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_2, LOW);
  Time_2 = pulseIn(echo_2, HIGH);
  distanceCM_2 = Time_2 * 0.034;
  return resultCM_2 = distanceCM_2 / 2;
}

Float sensor_3(void)
{
  digitalWrite(trig_3, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig_3, LOW);
```

```
Time_3 = pulseIn(echo_3, HIGH);
distanceCM_3 = Time_3 * 0.034;
return resultCM_3 = distanceCM_3 / 2;
```

ESP8266 Wi-Fi module:

The ESP8266 Wi-Fi module is used to send the data that collected by ultra Sonic sensor.

In this process we first collect the data from ultra Sonic sensor and this to mobile using the remote server ie. Cloud which can helps to get the data from the sensors.

For sending the data we use wifi connection to give the data to user via cloud server .

We must give the name of the channel of cloud server to send the information about empty spaces.

The code can be in any language like C, Python, Java, PHP etc

For my project i choose the code below :

```
#include “ThingSpeak.h”
#include <ESP8266WiFi.h>

Char ssid[] = “SSID”; //SSID here
Char pass[] = “PASSWORD”;
```

```
Unsigned long Channel_ID =123456;

Const char * myWriteAPIKey = “ACBDE12345”;

Const int Field_Number_1 = 1;
String value = “”;
Int value_1 = 0;
WiFiClient  client;

Void setup()
{
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
  Internet();
}

Void loop()
{
  Internet();
  If (Serial.available() > 0)
  {
    Delay(100);
    While (Serial.available() > 0)
    {
      Value = Serial.readString();
      If (value[0] == ‘*’)
```

```
    {
      If (value[2] == '#')
      {
        Value_1 = value[1] – 0x30;
      }
    }
  }
 }
 Upload();
}

Void internet()
{
 If (WiFi.status() != WL_CONNECTED)
 {
  While (WiFi.status() != WL_CONNECTED)
  {
   WiFi.begin(ssid, pass);
   Delay(5000);
  }
 }
}

Void upload()
{

 ThingSpeak.writeField(Channel_ID, Field_Number_1, value_1,
myWriteAPIKey);
```

```
  Delay(15000);
  Value = “”;
}
```

Cloud server:

For sending the parking place status data to mobile app we use cloud server.

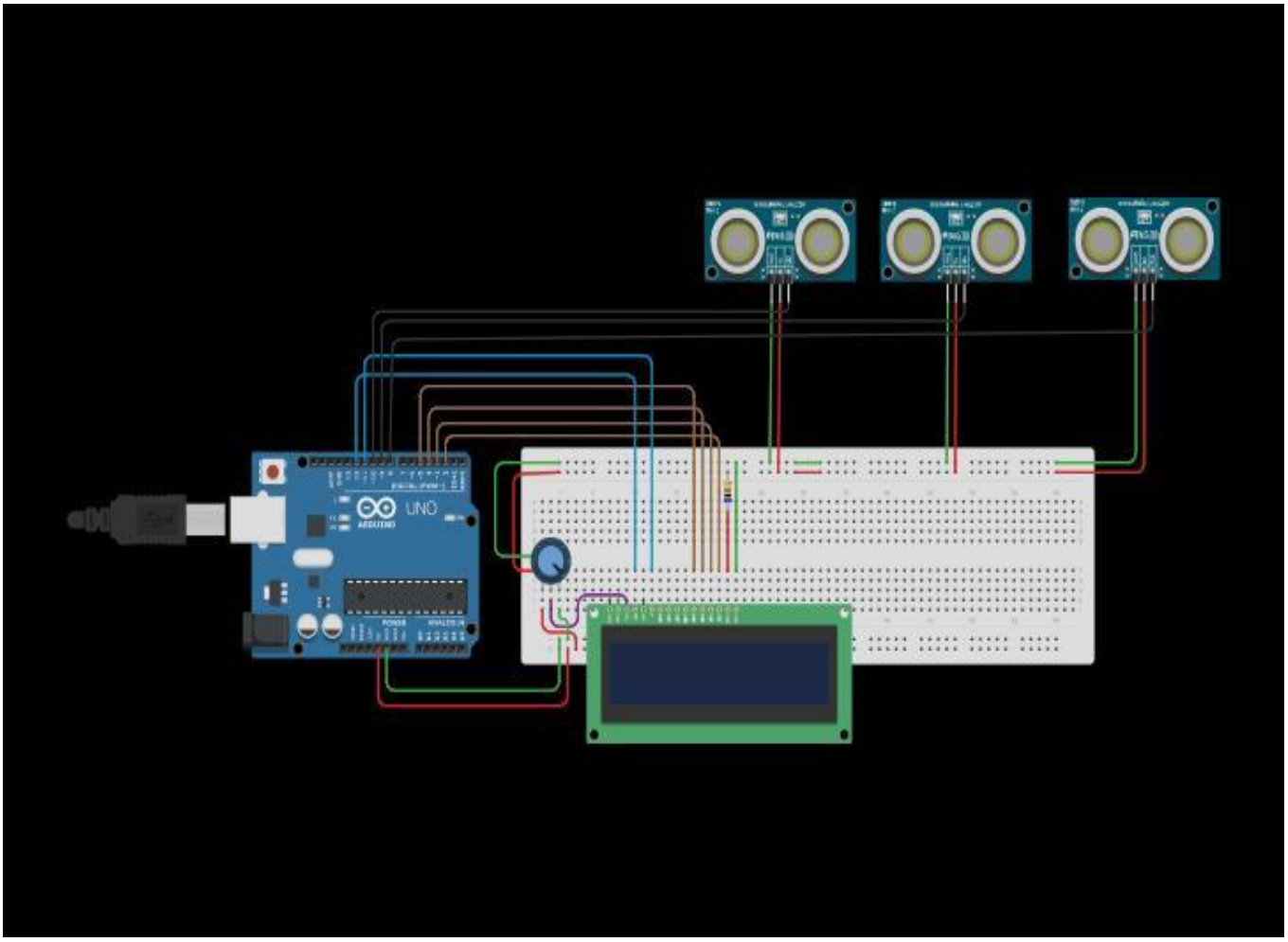
For our project we choose Thinkspeak cloud to send the information which is free for use.

For using Thinkspeak we must first have a account in the server to have the cloud server for use.

We can setup our channel in cloud as public to update the details about the parking spaces .

Now people can use the data about parking slot in the public tab using Thinkspeak cloud server.

Circuit diagram:



Code:

Python

```
# Import necessary libraries
Import time
Import RPi.GPIO as GPIO
Import requests

# Set GPIO mode and pin numbers
GPIO.setmode(GPIO.BCM)
```

```
Sensor_pin = 18

# Set API endpoint for sending data
Api_endpoint = https://your-api-endpoint.com/parkingsensor

# Initialize GPIO pin as input
GPIO.setup(sensor_pin, GPIO.IN)

# Function to send sensor data to the API
Def send_data(status):
   Data = {
      “status”: status,
      “timestamp”: int(time.time())
   }
   Response = requests.post(api_endpoint, json=data)
   If response.status_code == 200:
      Print(“Data sent successfully!”)
   Else:
      Print(“Failed to send data.”)

# Main loop
While True:
   Try:
      # Read sensor status (0 or 1)
      Status = GPIO.input(sensor_pin)

      # If parking spot is occupied, send status as 1
      If status == 1:
```

```
            Send_data(1)

        # If parking spot is vacant, send status as 0
        Else:
            Send_data(0)

        # Wait for 1 second before reading again
        Time.sleep(1)

    # Exit the program on keyboard interrupt (Ctrl+C)
    Except KeyboardInterrupt:
        Print("Program terminated.")
        GPIO.cleanup()
```

App development:

For developing a app for notifying the free space in parking area .our team created

A app through MIT inventor which gives simple and needable things for developing the app.

Steps for App developing in MIT inventor:

For these first go to MIT inventor website and click the create apps

Secondly we can open a new project to do our work

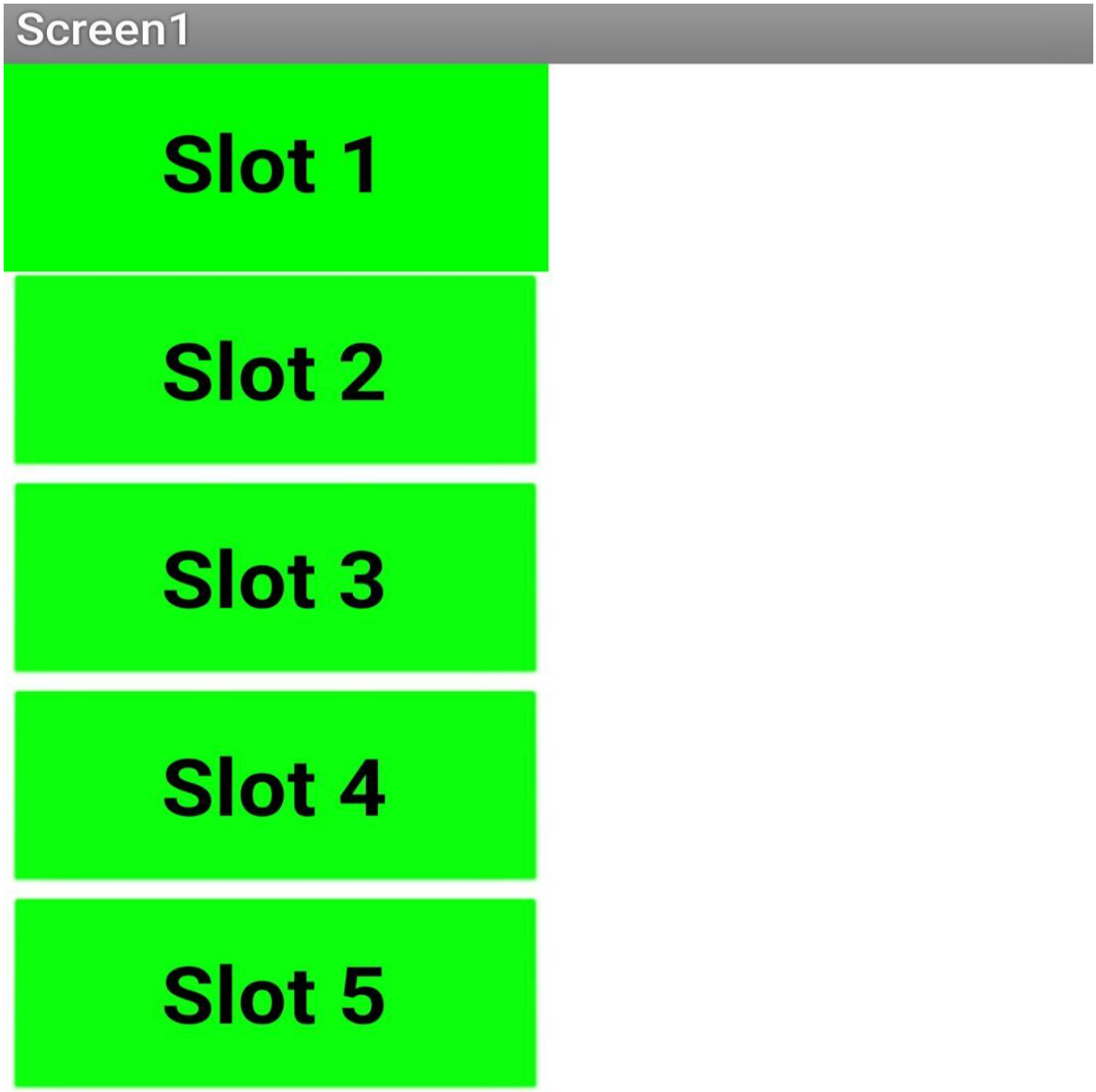
Then we can see the phone screen in our laptop screen .we can drag and drop the buttons or anything else that we need.

After designing the screen we must go to block section where we can drag the code that we want for the block to do in the app

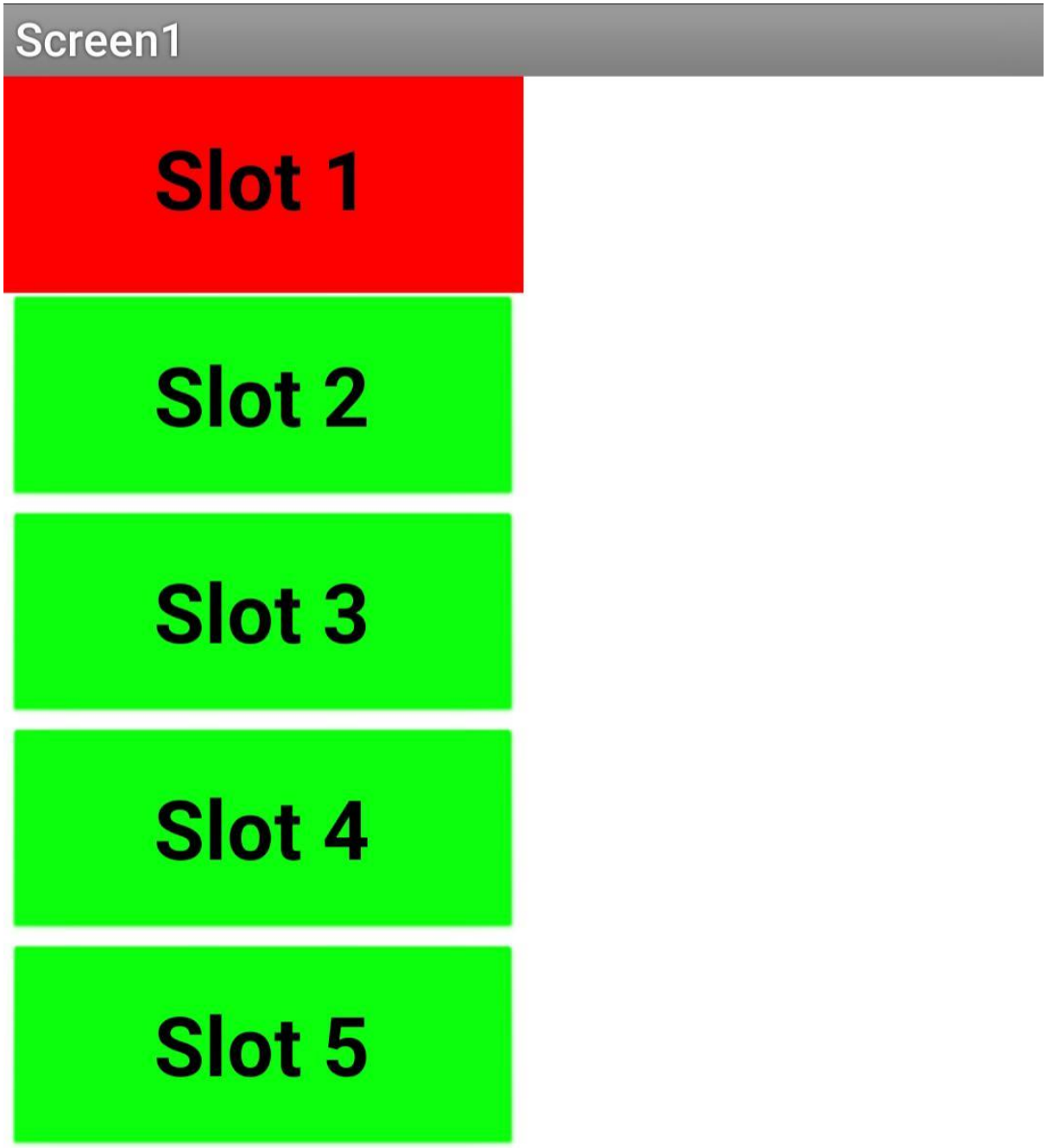
After this we can save the project and import into the computer.

Connect the project execution by several ways like wifi, USB etc

Scan the QR code and execute the project



Here the slot area is free to park we can choose any place we want



Here the slot 1 is not available for parking so we can search for nearby slot in parking area.

My app link is:
https://drive.google.com/file/d/1_ZPKt8vn4X8Mgd20aVASjN9nPrpfXpLU/view?usp=drivesdk

Conclusion

We conclude that creating a simple app for our project smart parking to search the slot for parking.we use ultrasonic sensor for sensing the slot is free or not. For app data we use thinkspeak which can helps to make app updated about parking slot.

Thank you