**CS 209 CS 210 PROJECT**

**SMART PARKING AUTOMATION**

**WITH VOICE CONTROLLED CAR PARKING**

By

T ANUDEEP (2101AI43)

P KARTHIK (2101AI21)

REPORT OF

* T ANUDEEP
* 2101AI43

**INTRODUCTION:**

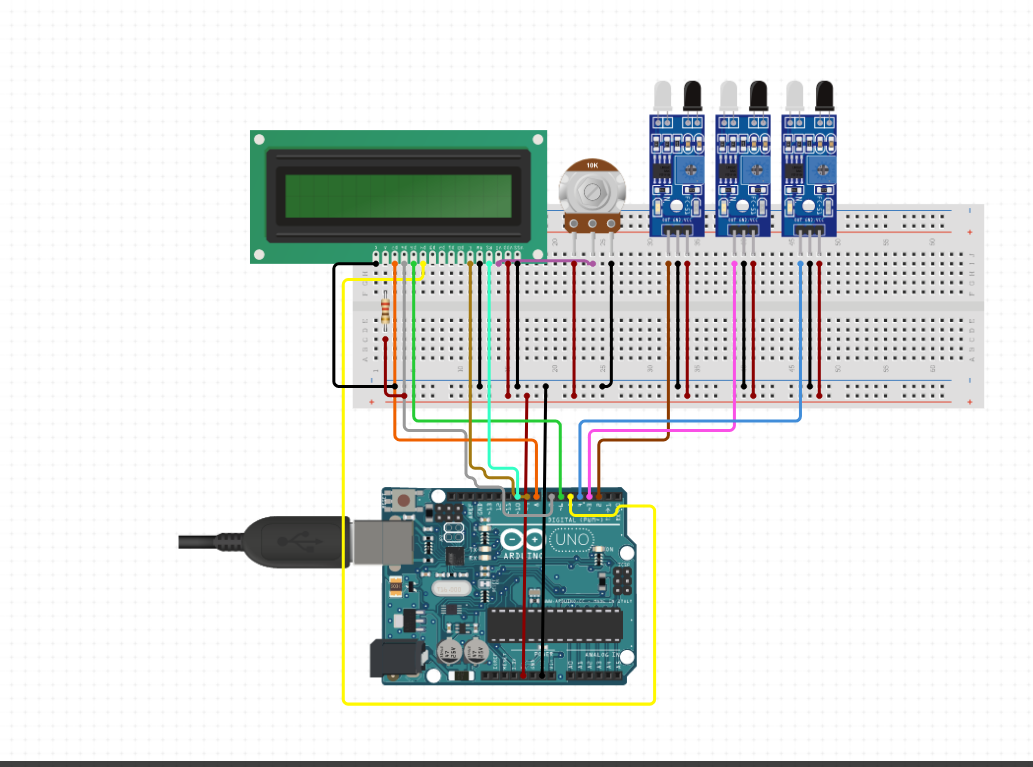
Nowadays, it is very difficult to park a car in parking slots due to irregular parking patterns by the drivers. Also it takes time to search for a parking slot and park there. To avoid this, we see a few commercial complexes, airports, malls etc. employ valet parking. This is to reduce commuter traffic. But even then, we see the traffic builds up if the number of valet parkers do not match the incoming traffic. Also, the employment charges are additional to the companies hiring. To overcome these problems and automate the parking system, we have created a SMART PARKING AUTOMATION WITH VOICE CONTROLLED CAR PARKING using open source hardware Arduino.

The project works as follows:

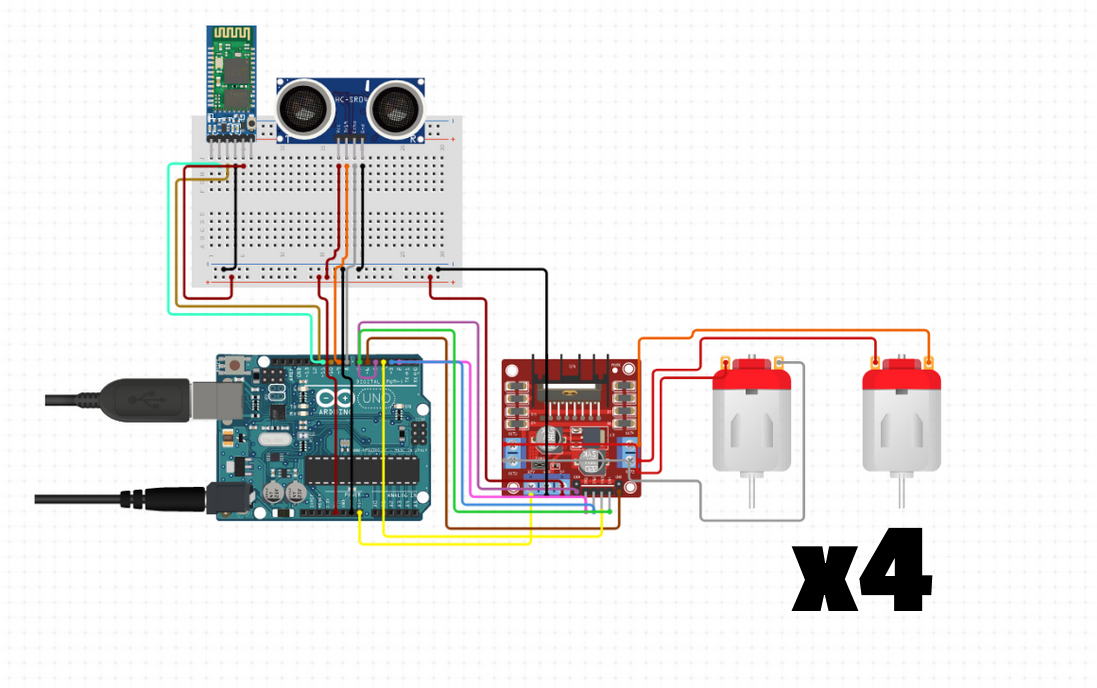
1. Creating a parking module which has a LCD display, showing the slots available. It displays whether the slot is full or empty.
2. Based on the empty slot shown, the user will choose an available slot. He will give the slot number as the voice input to an app, which we have made using MIT App Inventor.
3. After giving the input, the Bluetooth module on the car detects the input and parks the car in the designated slot.
4. If there is a car already parked in the slot and the user has given a that slot as input, then the car will stop in its way and avoid a collision.

**CIRCUIT DIAGRAM**

**PARKING MODULE**



**CAR**



**CODE FOR PARKING MODULE**

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 1);

void setup() {

  // set up the LCD's number of columns and rows:

  lcd.begin(16, 2);

  // Print a message to the LCD.

  lcd.print("   PARKING   ");

  pinMode(6,INPUT);

  pinMode(7,INPUT);

  pinMode(8,INPUT);

  pinMode(9,INPUT);

}

void loop() {

 // set the cursor to column 0, line 1

  // (note: line 1 is the second row, since counting begins with 0):

  lcd.setCursor(0, 1);

  if(digitalRead(6)==1){

  // print the number of seconds since reset:

  lcd.print("S1X");

  }

  else

  {

    lcd.print("S1F");

  }

  lcd.setCursor(4, 1);

  if(digitalRead(7)==1){

  // print the number of seconds since reset:

  lcd.print("S2X");

  }

  else

  {

    lcd.print("S2F");

  }lcd.setCursor(8, 1);

  if(digitalRead(10)==1){

  // print the number of seconds since reset:

  lcd.print("S3X");

  }

  else

  {

    lcd.print("S3F");

  }lcd.setCursor(12, 1);

}

**COMPONENTS USED:**

**FOR PARKING MODULE:**

1. Arduino UNO
2. Jumper wires
3. Breadboard
4. 16x2 LCD Display
5. 3xIR Sensors
6. Potentiometer

**FOR CAR**

1. Chassis set with attached wheels and motors
2. Arduino UNO
3. L298N Motor Driver
4. Breadboard
5. Jumper wires
6. HC-SR04 UltraSound sensor
7. HC05 Bluetooth Module

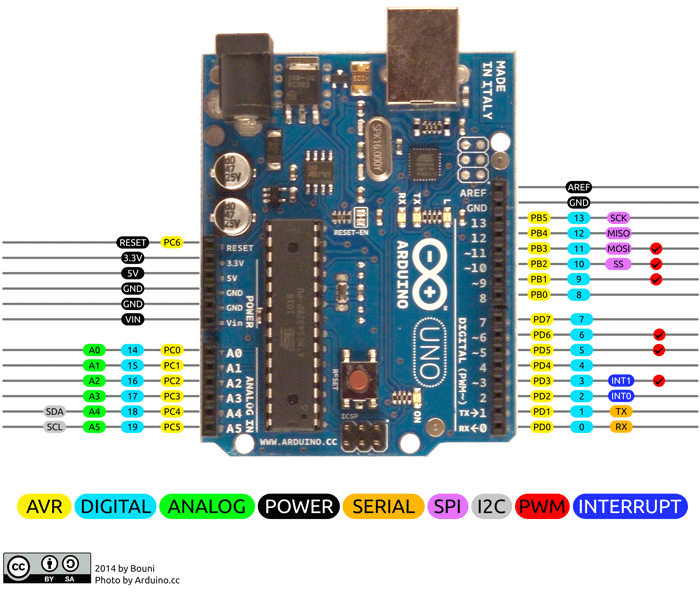
**INDIVIDUAL CONTRIBUTION**

In this project, I have worked on the setting up the parking module, its code and creating the app to give the voice input to the car.

**COMPONENTS OF PARKING MODULE:**

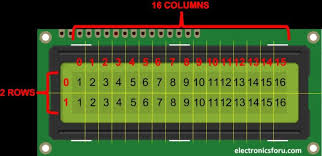
1. Arduino UNO

The Arduino Uno is an open-source microcontroller board based on the microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output pins. The board has 14 digital I/O pins(six capable of PWM output),6 analog I/O pins, and is programmable with the Arduino IDE via a type B USB cable. It can be powered by the USB cable or by an external 9V battery. Types of pins along with their names were shown in fig.



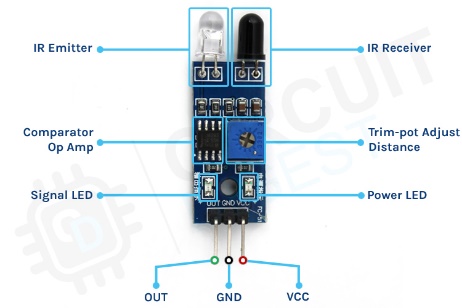
1. 16 x 2 LCD Display

A 16x2 LCD means **it can display 16 characters per line and there are 2 such lines**. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data. Command register stores various commands given to the display. Data register stores data to be displayed. Liquid Crystal Library simplifies this for you so you don't need to know the low-level instructions.



1. IR Sensors

An **infrared proximity sensor or IR Sensor** is an electronic device that emits infrared lights to sense some aspect of the surroundings and can be employed to detect the motion of an object. As this is a passive sensor, it can only measure infrared radiation. The **working of the IR sensor module** is very simple, it consists of two main components: the first is the IR transmitter section and the second is the IR receiver section. In the transmitter section, **IR led** is used and in the receiver section, a **photodiode** is used to receive infrared signal and after some signal processing and conditioning, you will get the output.



1. Jumper wires

Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Jumper wires come in three versions:

* Male-to-male jumper
* Male-to-female jumper
* Female-to-female jumper

And two types of head shapes: **square head** and **round head**.



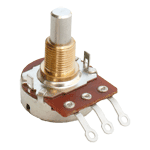
1. Breadboard

A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The breadboard has strips of metal underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally and split in the middle while the remaining holes are connected vertically.



6)Potentiometer

A potentiometer is a manually adjustable variable resistor with 3 terminals. Two of the terminals are connected to the opposite ends of a resistive element, and the third terminal connects to a sliding contact, called a wiper, moving over the resistive element. The potentiometer essentially functions as a variable resistance divider.

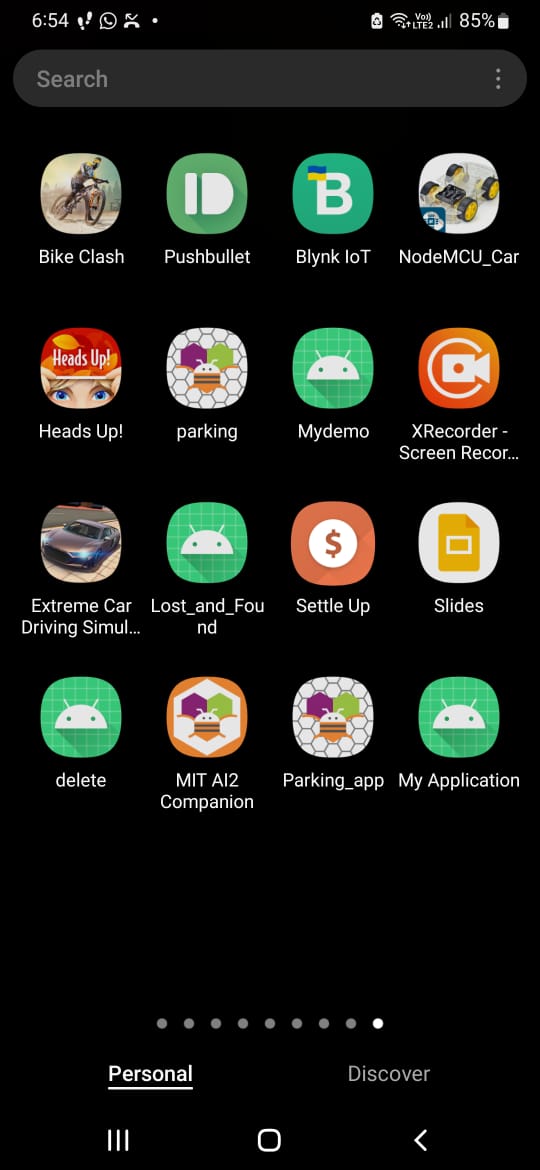


**WORKING OF PARKING MODULE**

The IR Sensors are placed at the three slot’s back end. The IR sensors give values of 0 if empty and 1 if full. Initially, the LCD display will show S1\_ S2\_ S3\_, where the \_ will be filled as follows. If the IR sensor reads 0 (is empty) at slot 1 (say), then the LCD will display S1X. If slot 1 is full(reads 1), then the LCD will display S1F. Similarly, for other slots, we get the similar readings.

**PARKING APP**

Using MIT App Inventor, I have made a parking app, which takes voice input. The app works as follows:



**Frontend:**

First open the app.



Then we have to connect to Bluetooth. Click on Connect. You can see the list of devices which are paired to Bluetooth. Select HC05.



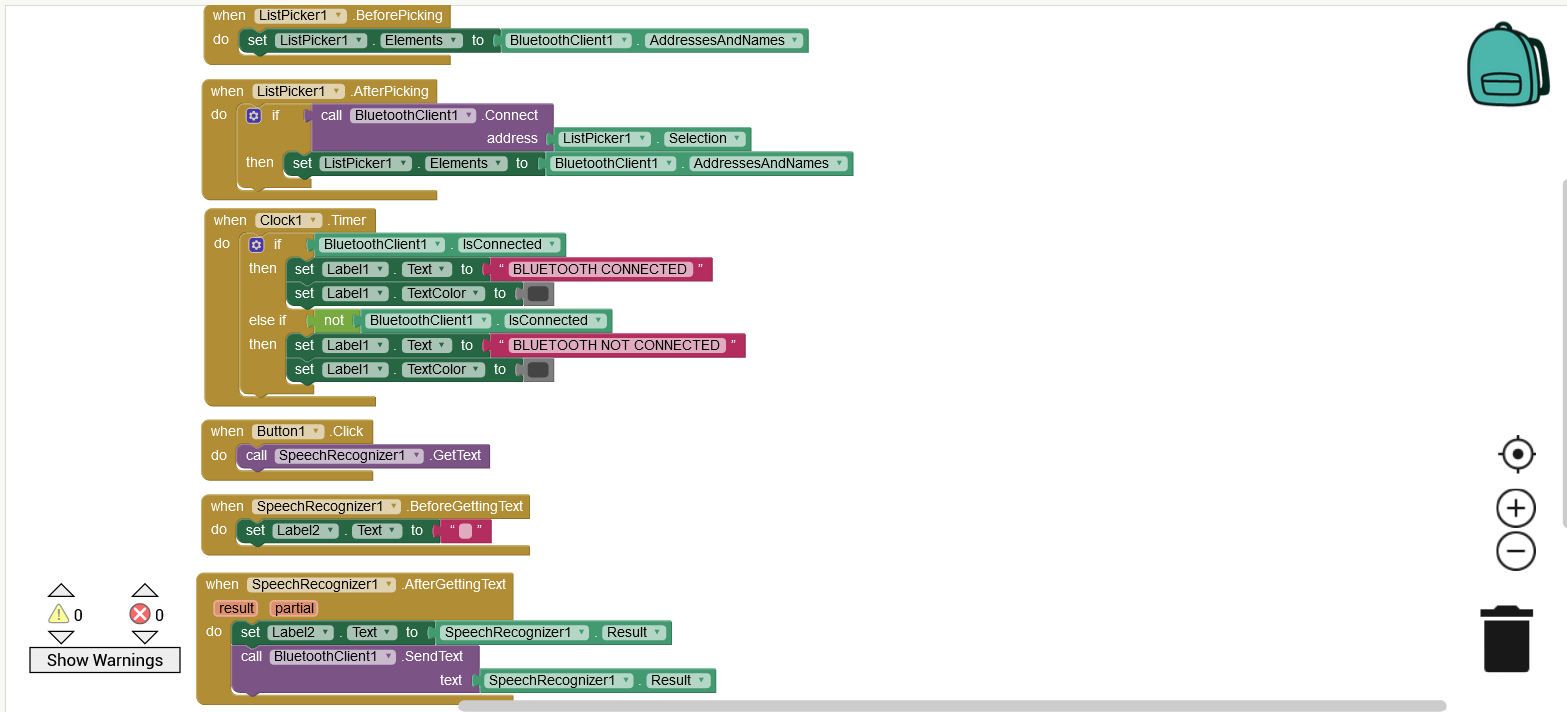
If it is connected successfully, then you can see BLUETOOTH CONNECTED. After that, click on voice input. Then Google Speech Services opens, which converts speech to text.



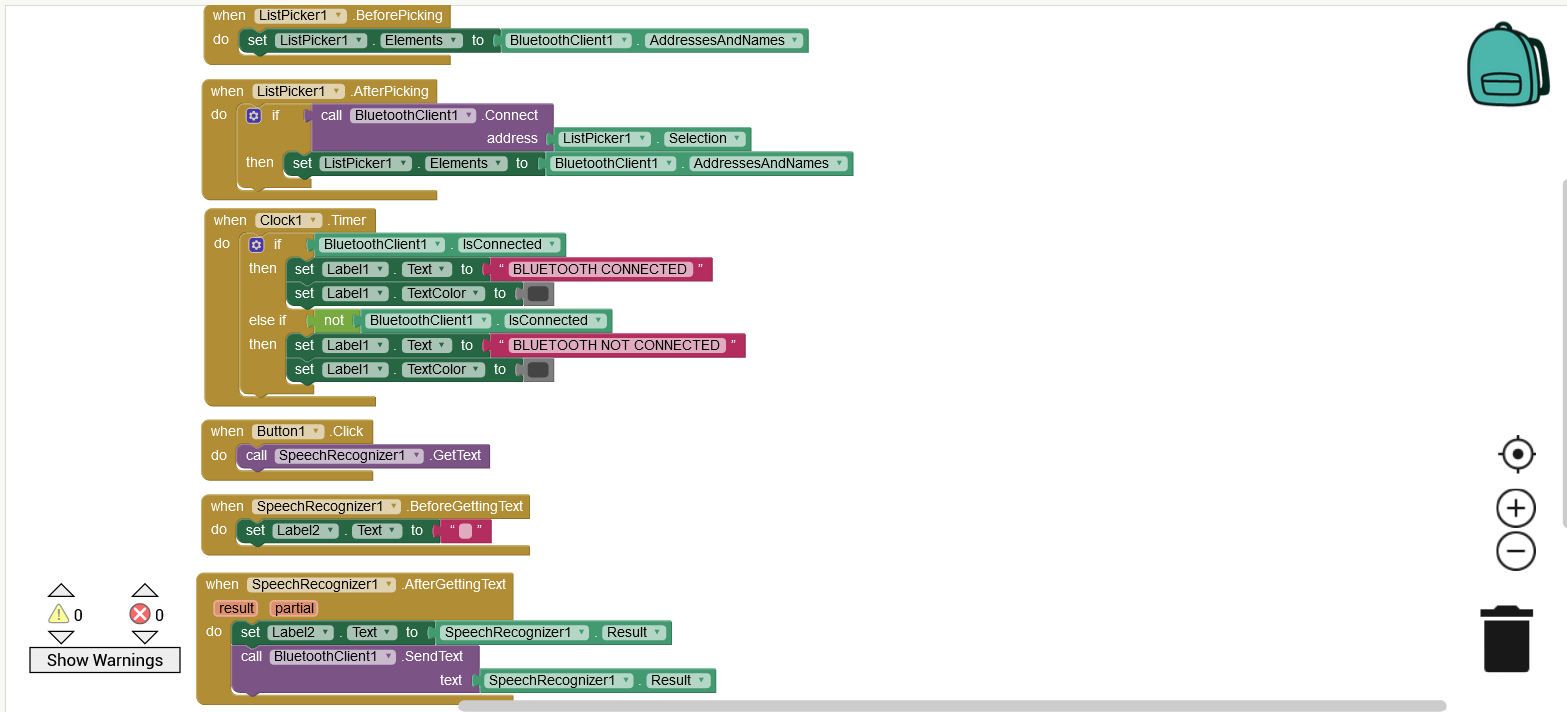
The voice input is given as follows:

Suppose slot one is empty and we want to park in slot one. Then say “one”. This input is passed to Bluetooth module and the car is parked in Slot 1. Sometimes, the Google Speech Services cannot recognize numerical inputs properly. If this problem occurs, we give input as individual letters of number – like “o n e”.

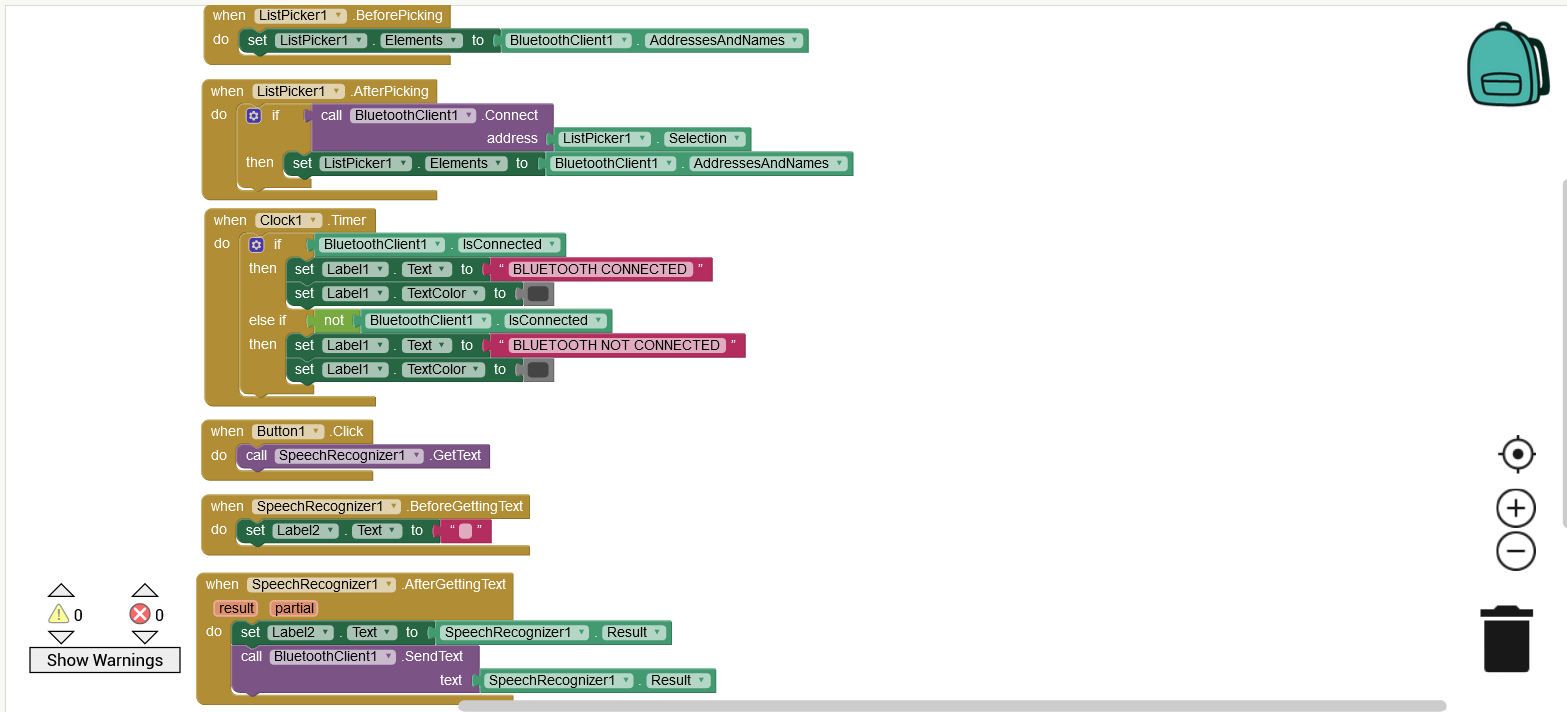
**Backend:**



**EXPLANATION OF BACKEND CODE**

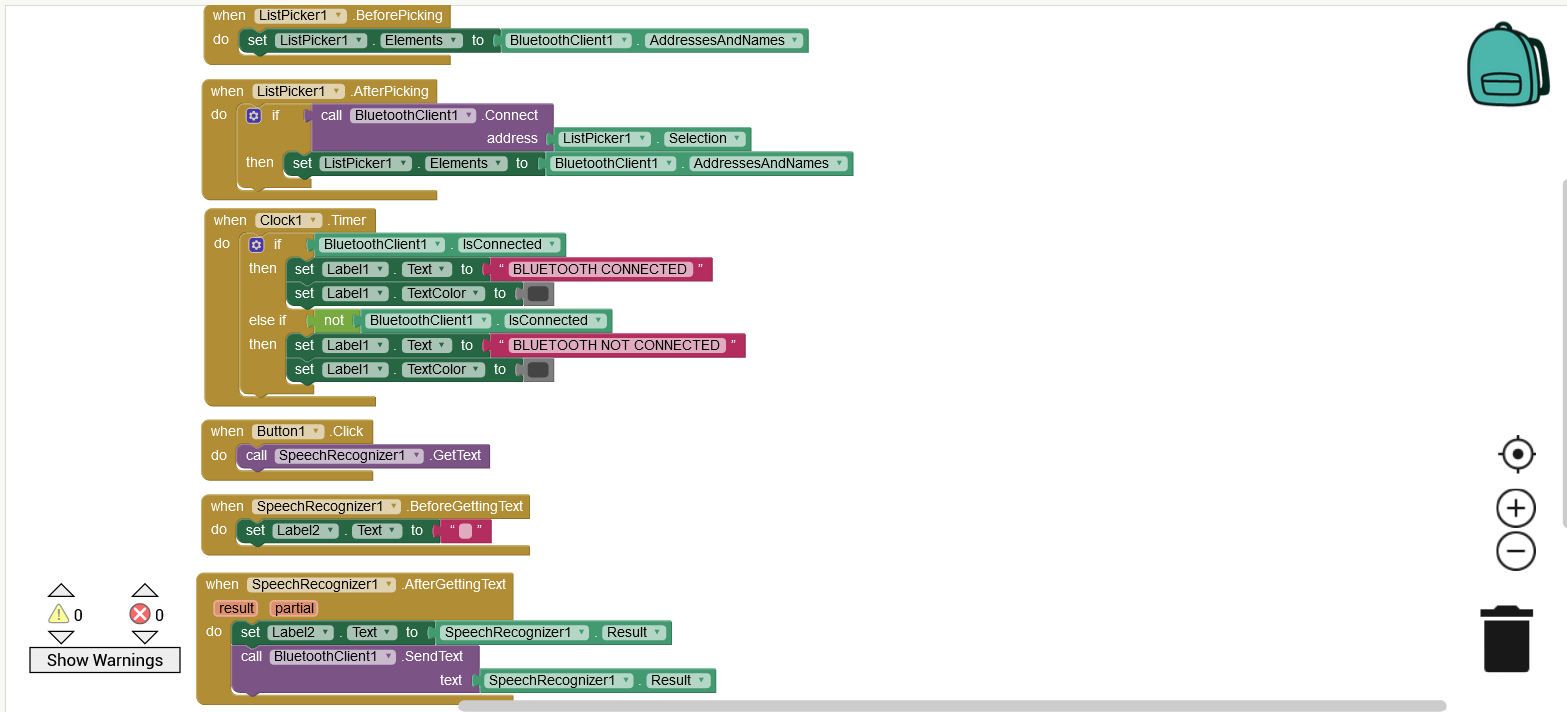


This block is used to validate the list picker, in which we have included the Connect button.

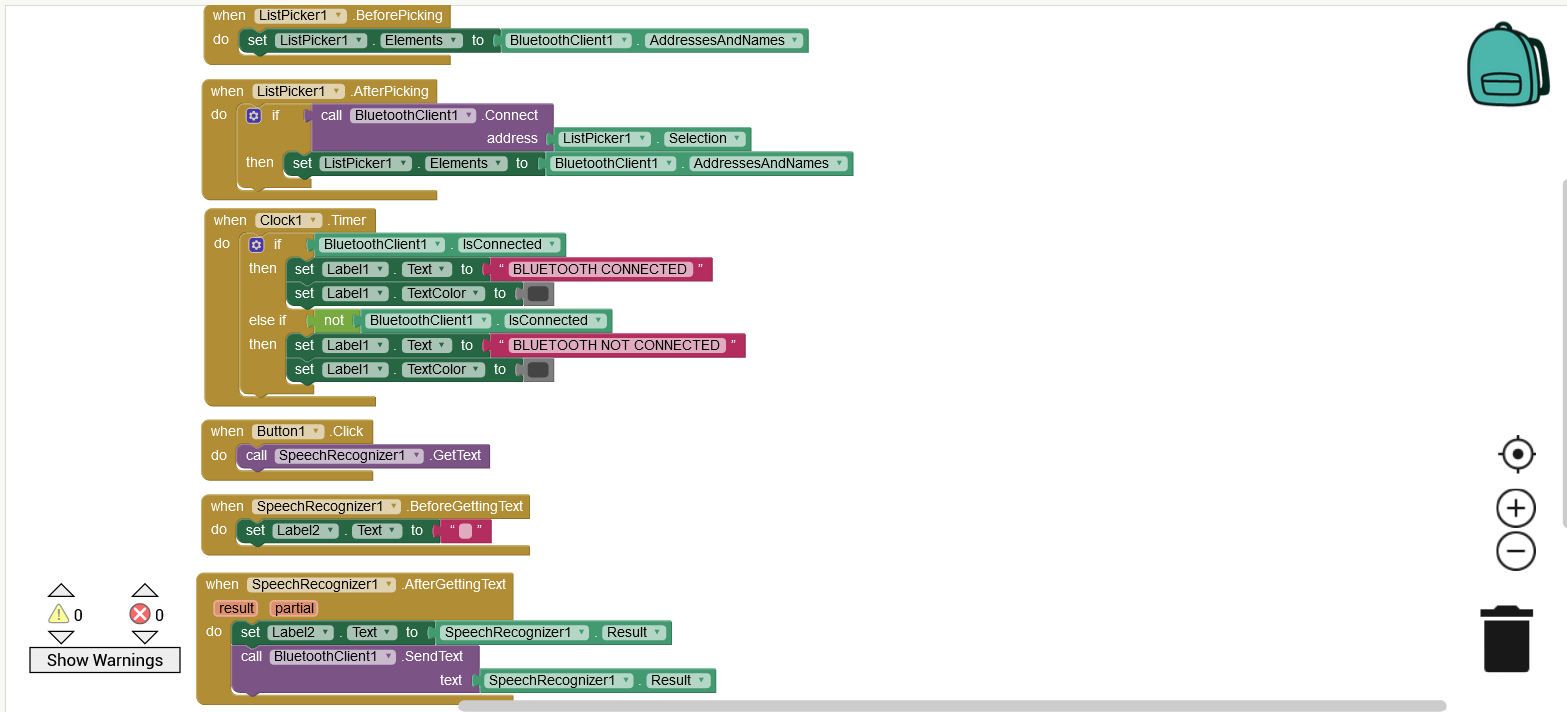


This block is used to validate what happens after clicking Connect button. It displays the list of all devices paired to Bluetooth in a new screen as follows

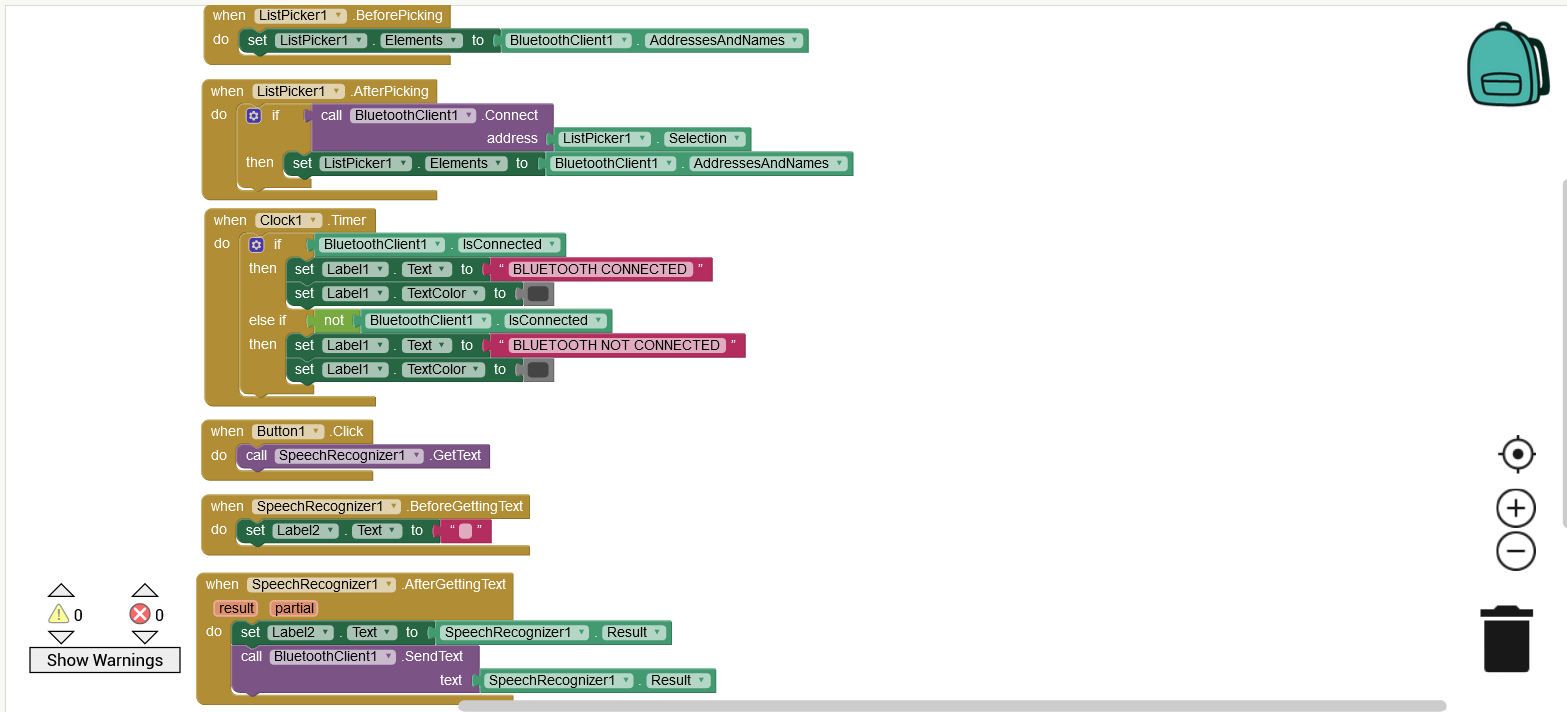




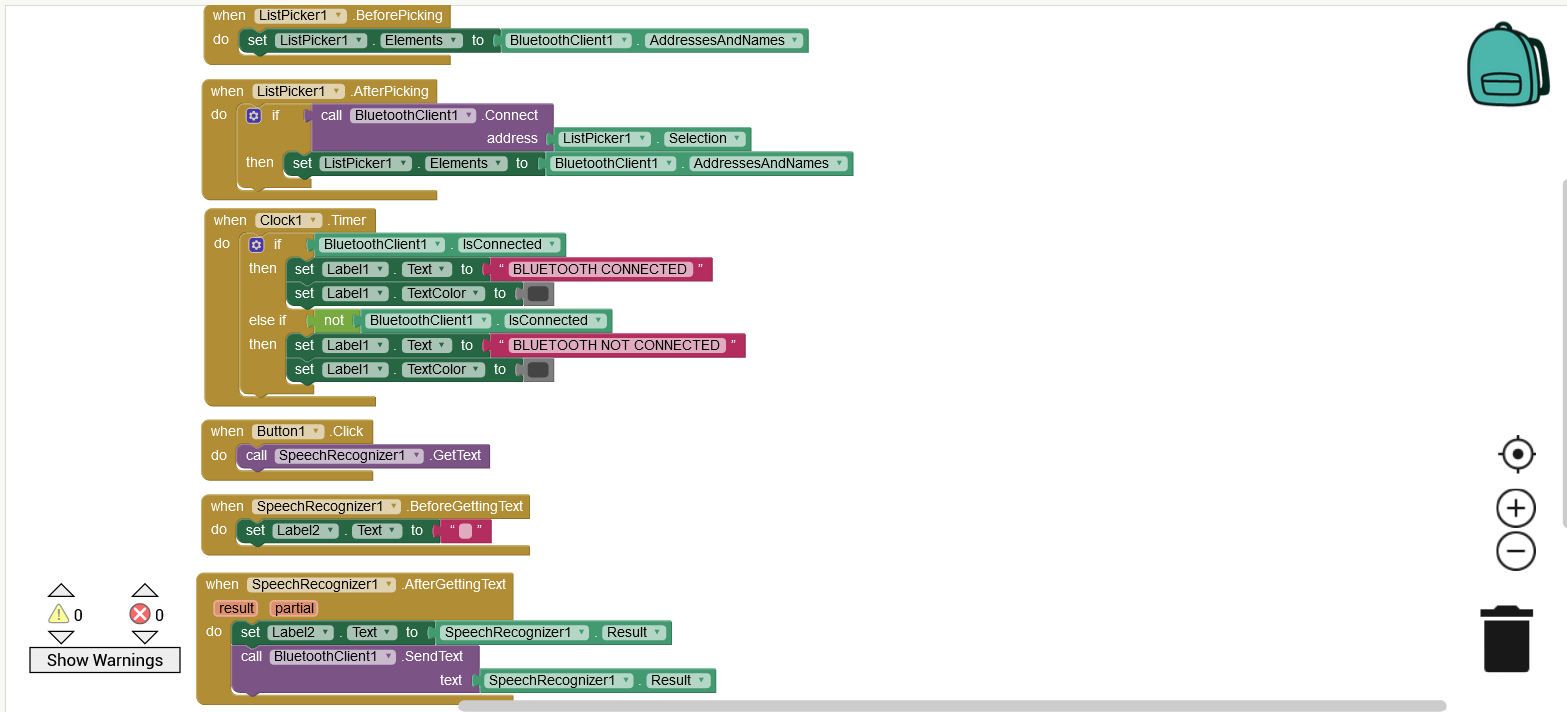
This block is used to connect to Bluetooth. It shows Bluetooth Connected on Main Screen if successfully connected. Else it shows Bluetooth Not Connected.



This block is used to validate Google Speech Services which converts speech to text.



This block takes the voice input



This block is used to send the voice input to the Bluetooth device (HC05) which is connected to the app.

**CONCLUSION**

In today’s world, where every task is moving towards automation, our project would serve as a correct replacement for a manual parking module. It is easy to use and requires very minimal technical skills for the driver. It can also avoid accidents owing to the threshold distance which we have incorporated in the car code. Such a model built on a large scale will cut down parking hardships to quite an extent.