SM Project Report

Smart Car Parking System

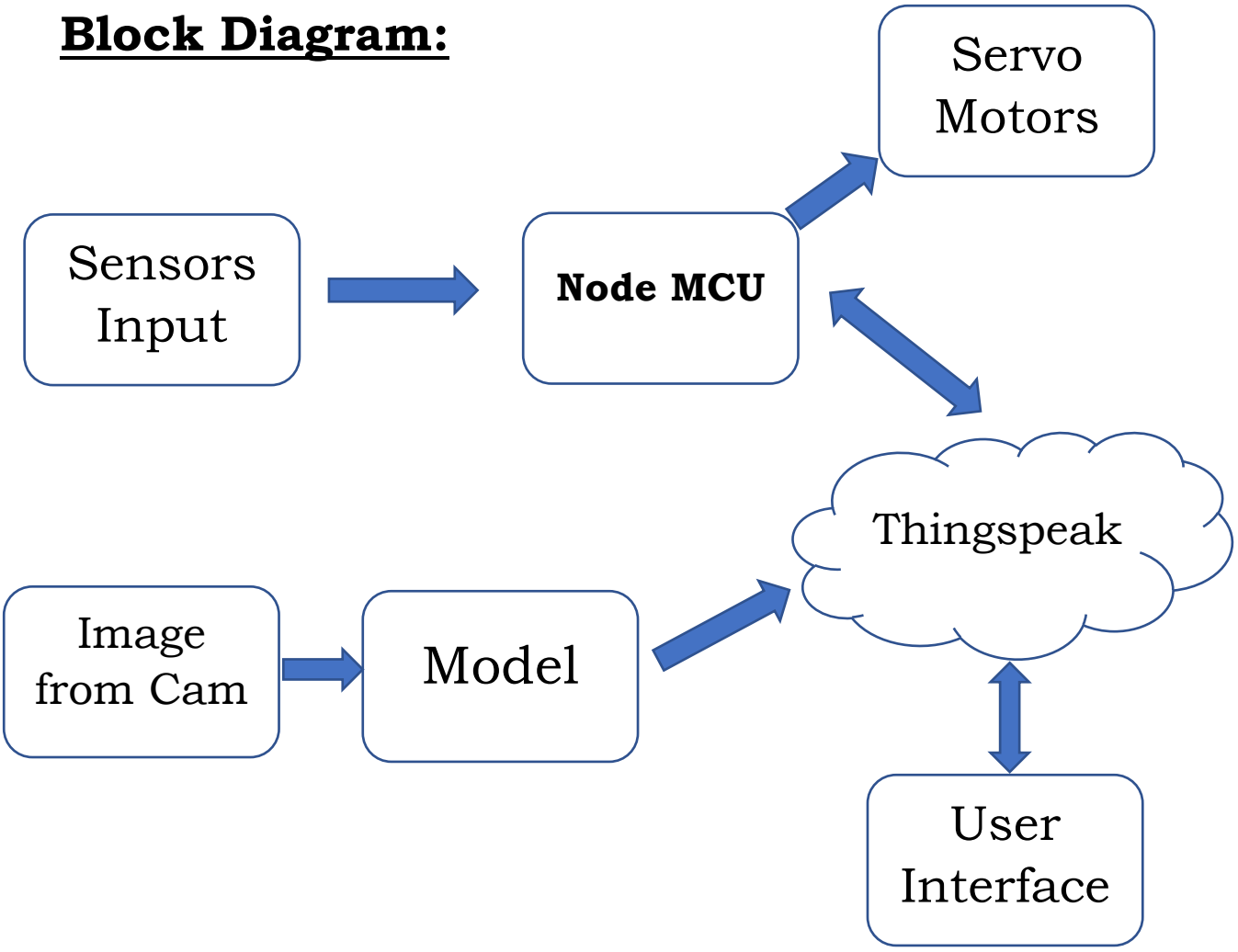
Group Details

D Mabu Jaheer Abbas - S20190020209
Thota Datta Aneesh - S20190010181
Thorlikonda Lakshmi Sai Gopinadh - S20190010180
Veduruparthi Sai Bhaskar - S20190010188

Abtract:

These days finding a parking slot is very difficult in major cities due to large vehicle population and lack of parking place. So smart parking system has been implemented, but these only monitor the local areas in metropolitan cities. So, we want to implement a system that can collect data of all parking slots in a city and can enable then to book by an people from the user interface, cloud will mediate between user interface and hardware systems. The main part we are implementing is enabling to book parking slot in the given city, this is the one which is novel in our project.

Block Diagram:



Functional Block Diagram connecting the sub-modules of the project

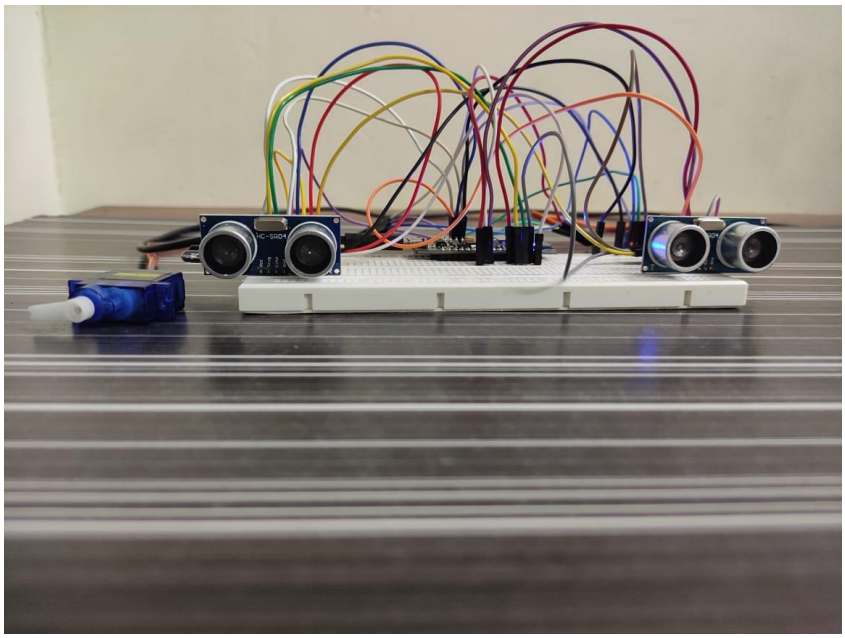
Hardware Modules:

- NodeMCU
- Servo Motors
- IR Sensors
- LEDs
- Ultrasonic Sensors
- Network Requirements
- Wifi (NodeMCU as wifi module)

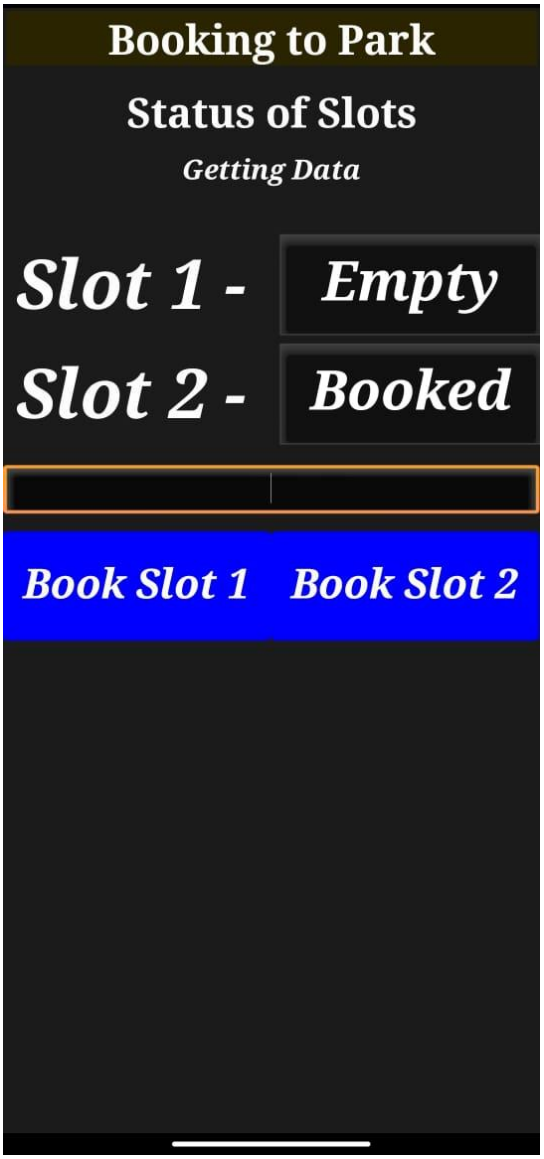
Software Modules:

- ThingSpeak cloud

- MIT App Inventor (to build App)



App which we have build



App output when system is working

Code:

NodeMCU

```
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include "ThingSpeak.h"

#include <Servo.h>

#define ir1 16 //D0
#define ir2 5  //D1
#define ultra1_echo 4 //D2
#define ultra1_trig 0 //D3
#define ultra2_echo 2 //D4
#define ultra2_trig 14 //D5
#define servopin 12 //D6
#define servopin2 13 //D7

Servo myservo;
Servo myservo2;
int sensor1, ulsensor1;
int sensor2, ulsensor2;

String slot1;
String slot2;
String cv, book1, book2;

// Ensure correct credentials to connect to your WiFi Network.
char ssid[] = "Abbas-mi";
char pass[] = "12345678";

// Ensure that the credentials here allow you to publish and subscribe to the ThingSpeak channel.
```

detect.py | license_plate_recognizer.py | utils.py

C: > Users > JAHEER ABBAS > Desktop > yolov4-custom-functions > license_plate_recognizer.py

```python
# test file if you want to quickly try tesseract on a license plate image
import pytesseract
import cv2
import os
import numpy as np

# If you don't have tesseract executable in your PATH, include the following:
# pytesseract.pytesseract.tesseract_cmd = r'<full_path_to_your_tesseract_executable>'
# Example tesseract_cmd = r'C:\Program Files (x86)\Tesseract-OCR\tesseract'

# point to license plate image (works well with custom crop function)
gray = cv2.imread("./detections/crop/car3/license_plate_.png", 0)
gray = cv2.resize( gray, None, fx = 3, fy = 3, interpolation = cv2.INTER_CUBIC)
blur = cv2.GaussianBlur(gray, (5,5), 0)
gray = cv2.medianBlur(gray, 3)
# perform otsu thresh (using binary inverse since opencv contours work better with white text)
ret, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
cv2.imshow("Otsu", thresh)
cv2.waitKey(0)
rect_kern = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))

# apply dilation
dilation = cv2.dilate(thresh, rect_kern, iterations = 1)
#cv2.imshow("dilation", dilation)
#cv2.waitKey(0)
# find contours
try:
    contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
except:
    ret_img, contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
sorted_contours = sorted(contours, key=lambda ctr: cv2.boundingRect(ctr)[0])

# create copy of image
im2 = gray.copy()

plate_num = ""
```

Challenges:

- Could not detect number plates which is not in proper format. For example, if the number plate is broken at some corner, it will be hard to detect for deep learning model and extract the number from number plate.
- If someone booked the parking slot and did not show up then the system does not know what to do. It will be a challenge because it books the slot which make others could not book the slot.