**PHASE-5**

**Objective :**

Smart car parking project aims at providing a confusion free and easy parking. This project helps the drivers of the cars to park their vehicles with minimum wastage of time with accurate information of the availability of the space to park It includes an Arduino Uno as the microcontroller unit to which the servo motors, LCD display ultrasonic sensors (HC-05) are interfaced. The LCD displays the availability of the space, the ultrasonic sensors keeps the check of the number of cars entering and exiting the parking space. The ultrasonic sensors detect the availability of the parking space.

**THEORY**

An automated car parking system is a process through which car parking can be done more efficiently and easily than manual method. The system will provide the user better services.

The system counts the number of cars in the space and checks if there’s any vacancy. There’s an entry and exit path. When vehicle enters, the display shows the number of cars inside. When any vehicle leaves, the count decreases and shown on display. If the space is full. The display will show a message regarding that.

This whole process includes the use of Arduino, Display and sonar. The sonar detects whether the vehicle is entering or leaving. The report then showed on display.

**Hardware Design**

Hardware equipment that we need in order to build the project are given below:

1) Arduino UNO (1 Nos)

2) Ultrasonic Sensor (3 Nos)

3) IR Sensor (1 Nos)

4) LCD display (1 Nos)

5) Servo Motor (1 Nos)

6) Red Led (3 Nos)

7) Green Led (3 Nos)

**Block Diagram (Sensor Setup):**

Block Diagram of our proposed system is given below :

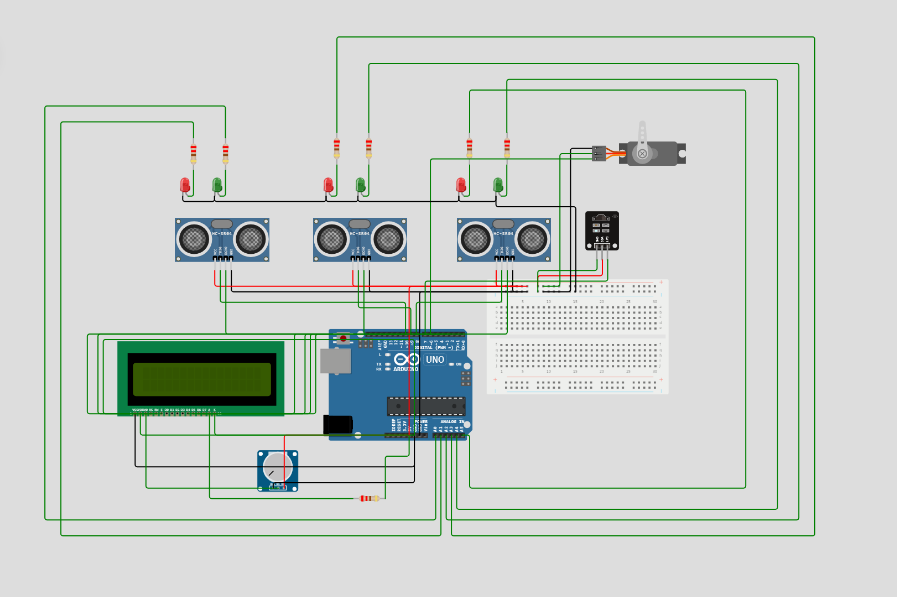
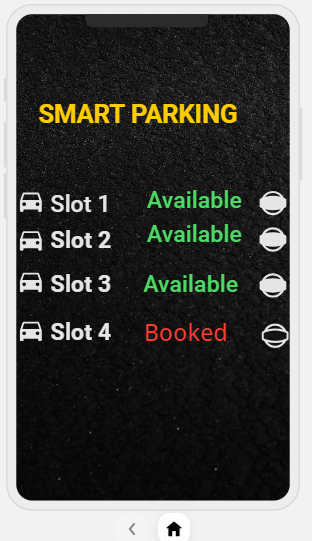


Figure- 1 : Block diagram of Smart Car Parking System



**Code Implementation**

#include <LiquidCrystal.h>

#include <Servo.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define t1 10

#define t2 9

#define t3 8

#define IR\_SENSOR\_PIN 7

#define SERVO\_PIN 6

#define GREEN\_LED1 A0

#define GREEN\_LED2 A2

#define GREEN\_LED3 A4

#define RED\_LED1 A1

#define RED\_LED2 A3

#define RED\_LED3 A5

Servo servoMotor;

int distanceThreshold = 150;

void setup() {

lcd.begin(16,2);

lcd.setCursor(0,0);

pinMode(IR\_SENSOR\_PIN, INPUT\_PULLUP);

pinMode(GREEN\_LED1, OUTPUT);

pinMode(GREEN\_LED2, OUTPUT);

pinMode(GREEN\_LED3, OUTPUT);

pinMode(RED\_LED1, OUTPUT);

pinMode(RED\_LED2, OUTPUT);

pinMode(RED\_LED3, OUTPUT);

servoMotor.attach(SERVO\_PIN);

Serial.begin (9600);

}

long readDistance(int triggerPin, int echoPin) {

pinMode(triggerPin, OUTPUT);

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

pinMode(echoPin, INPUT);

return pulseIn(echoPin, HIGH);

}

void handleLEDs(float distance, int greenLedPin, int redLedPin) {

if (distance >= distanceThreshold){

digitalWrite(greenLedPin, HIGH);

digitalWrite(redLedPin, LOW);

Serial.println("Empty slot!!");

} else {

digitalWrite(redLedPin, HIGH);

digitalWrite(greenLedPin, LOW);

Serial.println("Car is parked");

}

}

void loop() {

//... same code as above ...

float d1 = 0.01723 \* readDistance(t1, t1);

float d2 = 0.01723 \* readDistance(t2, t2);

float d3 = 0.01723 \* readDistance(t3, t3);

Serial.println("d1 = " + String(d1) + "cm");

Serial.println("d2 = " + String(d2) + "cm");

Serial.println("d3 = " + String(d3) + "cm");

if (digitalRead(IR\_SENSOR\_PIN) == LOW) { // Vehicle detected by IR sensor

if (d1 > 100 || d2 > 100 || d3 > 100) {

servoMotor.write(90); // Assuming 90 degrees opens the gate

delay(500);

servoMotor.write(0); // Assuming 0 degrees closes the gate after some time

delay(500);

} else {

// Do nothing or maybe sound a buzzer to indicate parking full

}

}

// Handle LEDs based on the distance readings

handleLEDs(d1, GREEN\_LED1, RED\_LED1);

handleLEDs(d2, GREEN\_LED2, RED\_LED2);

handleLEDs(d3, GREEN\_LED3, RED\_LED3);

if (d1>100 & d2>100 & d3>100){

lcd.setCursor(0,0);

lcd.print("3 Slots Free");

lcd.setCursor(0,1);

lcd.print("Slot 1 2 3 Free");

delay(500);

}

else if((d1>100 & d2>100)|(d2>100 & d3>100)|(d3>100 & d1>100))

{

lcd.setCursor(0,0);

lcd.print("2 Slots Free");

lcd.setCursor(0,1);

if(d1>100 & d2>100)

lcd.print("Slot 1 & 2 Free");

else if(d1>100 & d3>100)

lcd.print("Slot 1 & 3 Free");

else

lcd.print("Slot 2 & 3 Free");

delay(500);

}

else if(d1<100 & d2<100 & d3<100)

{

lcd.setCursor(0,0);

lcd.print("No Slot Free");

lcd.setCursor(0,1);

lcd.print("Parking Full");

delay(500);

}

else if((d1<100 & d2<100)|(d2<100 & d3<100)|(d3<100 & d1<100))

{

lcd.setCursor(0,0);

lcd.print("1 Slot Free");

lcd.setCursor(0,1);

if(d1>100)

lcd.print("Slot 1 is Free");

else if (d2>100)

lcd.print("Slot 2 is Free");

else

lcd.print("Slot 3 is Free");

delay(500);

}

delay(100);

}

**Code Implementation (flutter):**

//build.gradle

buildscript {

repositories {

google()

jcenter()

}

dependencies {

classpath 'com.android.tools.build:gradle:3.2.1'

classpath 'com.google.gms:google-services:3.2.1'

}

}

allprojects {

repositories {

google()

jcenter()

}

}

rootProject.buildDir = '../build'

subprojects {

project.buildDir = "${rootProject.buildDir}/${project.name}"

}subprojects {

project.evaluationDependsOn(':app')

}

task clean(type: Delete) {

delete rootProject.buildDir

}

//app

def localProperties = new Properties()

def localPropertiesFile = rootProject.file('local.properties')

if (localPropertiesFile.exists()) {

localPropertiesFile.withReader('UTF-8') { reader ->

localProperties.load(reader)

}

}

def flutterRoot = localProperties.getProperty('flutter.sdk')

if (flutterRoot == null) {

throw new GradleException("Flutter SDK not found. Define location with flutter.sdk in the local.properties file.")

}

def flutterVersionCode = localProperties.getProperty('flutter.versionCode')

if (flutterVersionCode == null) {

flutterVersionCode = '1'

}

def flutterVersionName = localProperties.getProperty('flutter.versionName')

if (flutterVersionName == null) {

flutterVersionName = '1.0'

}

apply plugin: 'com.android.application'

apply from: "$flutterRoot/packages/flutter\_tools/gradle/flutter.gradle"

android {

compileSdkVersion 28

lintOptions {

disable 'InvalidPackage'

}

defaultConfig {

// TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).

applicationId "com.smart\_car\_parking"

minSdkVersion 16

targetSdkVersion 28

versionCode flutterVersionCode.toInteger()

versionName flutterVersionName

testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

multiDexEnabled true

}

buildTypes {

release {

// TODO: Add your own signing config for the release build.

// Signing with the debug keys for now, so `flutter run --release` works.

signingConfig signingConfigs.debug

}

}

}

flutter {

source '../..'

}

dependencies {

testImplementation 'junit:junit:4.12'

androidTestImplementation 'com.android.support.test:runner:1.1.1'

androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.1.1'

implementation 'com.android.support:multidex:1.0.3'

}

apply plugin: 'com.google.gms.google-services'

**Flow Chart**

The flow chart includes how the system works. The program flow chart is given below:

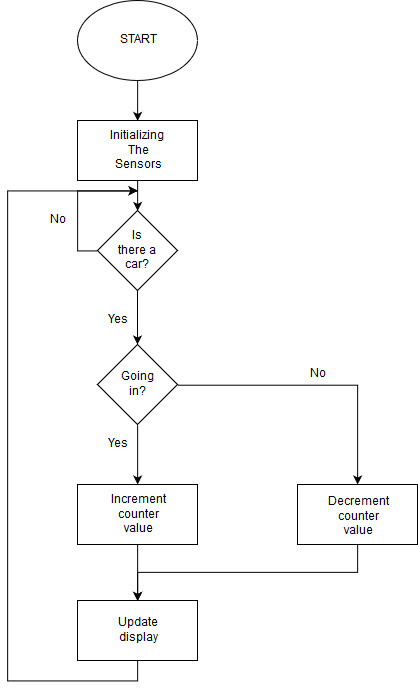


Figure-2 : Flow chart of Smart Car Parking System

**References**

1. <https://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet>
2. <https://store.arduino.cc/usa/arduino-uno-rev3>
3. <http://www.micropik.com/PDF/HCSR04.pdf>