Mapúa University

School of Electrical, Electronics and Computer Engineering

Automatic Car Parking System

Submitted by:

CENTINO, Mark Joshua

CHUA, Cyrille Lan

NOPIA, John Paul

ECE131L/B14

GROUP 2

ENGR. GLENN MAGWILI

January 11, 2020

PROJECT DESCRIPTION:

Car parking is a major problem in urban areas in both developed and developing countries. Following the rapid increase of car ownership, many cities are lacking car parking areas. This imbalance is partially due to ineffective land use planning and miscalculations of space requirements during first stage of planning. Shortage of parking space, high parking tariffs, and traffic congestion due to visitors in search for a parking place are only a few problems examples of everyday parking. Parallel parking is usually a driver's worst nightmare because, it not only requires the driver's skills but also increases the possibility of other drivers bumping into their parked vehicle. This project consists of Arduino Uno board which is the main microcontroller; IR sensors which detect objects infront of it, LEDs which indicates the availability of parking space wherein when lit means the space isn't occupied, LCD display which displays the status of the parking space and MATLAB GUI which projects the occupied parking spaces.

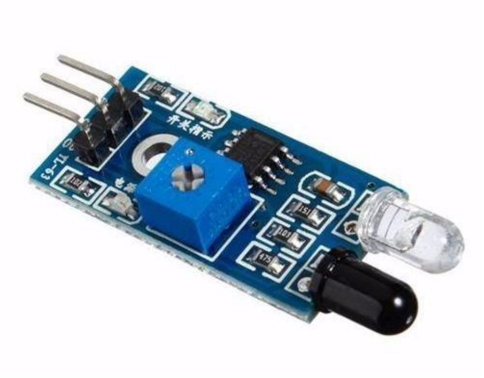
This project's main purpose is to produce a real-life solution to the car parking problem which the whole world is facing frequently. People usually roam around in the parking lots trying to find a suitable place to park in. To solve that problem, we have created the automatic car parking system, using an open source hardware, programmable sensors and the use of computers to provide an interface to understand the digital output produced.

Components and Compatibility

The basic components used in our system are as follows:

1) IR Sensors (x4)

An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called a passive IR sensor. Usually, in the infrared spectrum, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, that can be detected by an infrared sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.



2) Arduino Uno (x1)

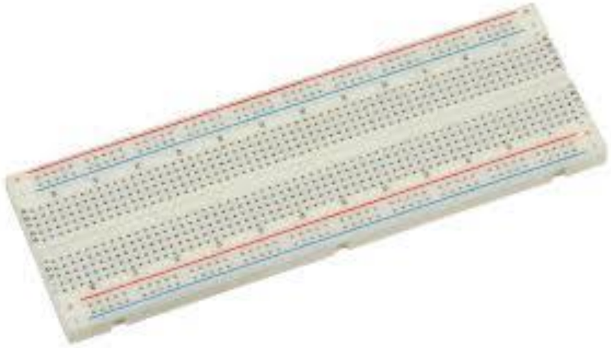
The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 6 analog inputs, 14 digital input/output pins (of which 6 can be used as PWM outputs), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; all we have to do is simply connect it to PC with a USB cable or power it with an AC-to-DC adapter or battery to get started.

3) Jump Wires (x40)

Wires are nothing but single, cylindrical, flexible strands of metal. They are usually used to bear loads of electric or telecommunication signals.

4) Breadboard (x1)

While building electric circuits, breadboards are the most fundamental pieces.



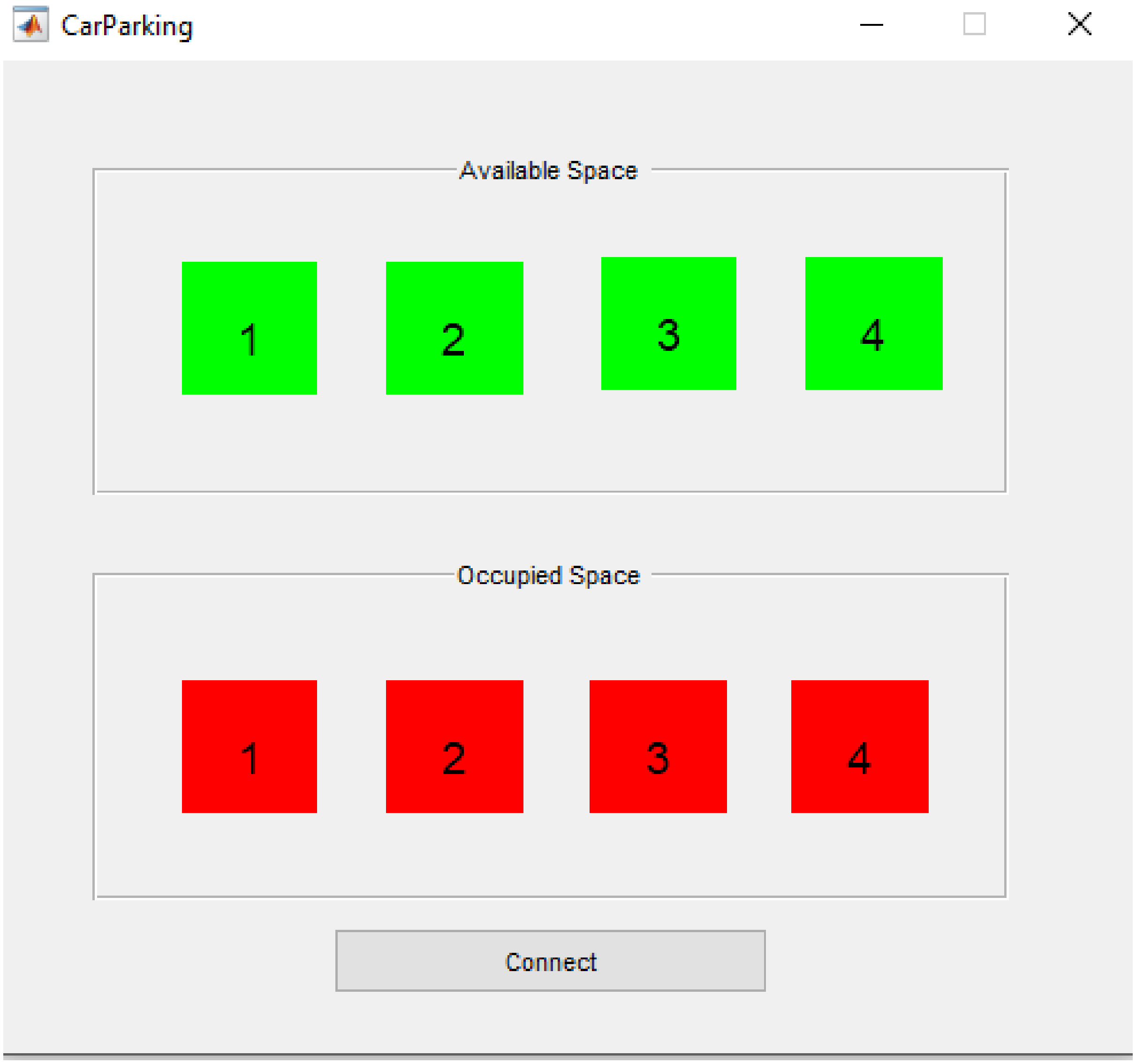
5) Resistors (x4)

Resistors are electronic components which have a specific, never-changing electrical resistance. The resistor's resistance limits the flow of electrons through a circuit. They are passive components, meaning they only consume power (and can't generate it).

6) LEDs (x4)

A light-emitting diode is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

MATLAB GUI


CarParking
Available Space
1
2
3
4
Occupied Space
1
2
3
4
Connect

Actual Photos of the Prototype

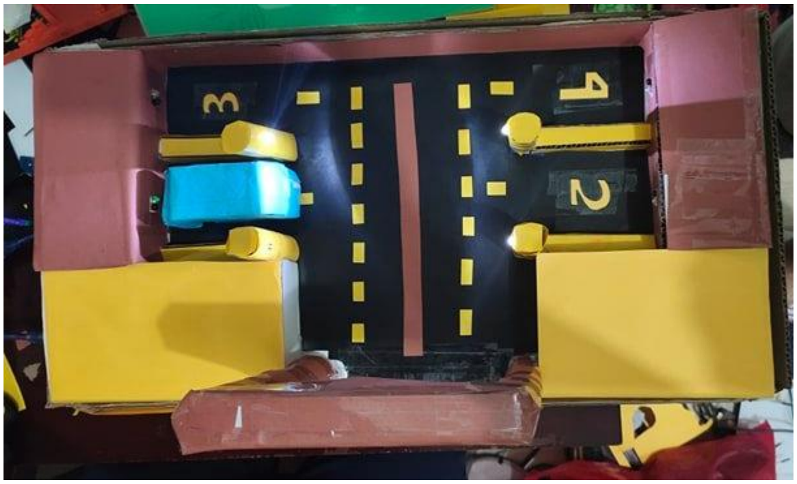


Figure 1. Bird's Eye View of the Prototype

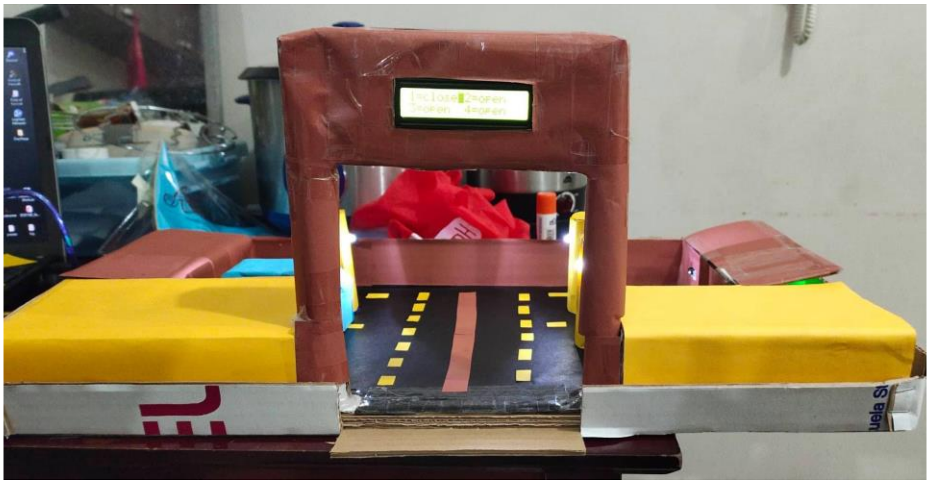


Figure 2. Front View of the Prototype

MATLAB CODE

```
function varargout = CarParking(varargin)
% CARPARKING MATLAB code for CarParking.fig
%      CARPARKING, by itself, creates a new CARPARKING or raises the existing
%      singleton*.
%
%      H = CARPARKING returns the handle to a new CARPARKING or the handle to
%      the existing singleton*.
%
%      CARPARKING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CARPARKING.M with the given input arguments.
%
%      CARPARKING('Property','Value',...) creates a new CARPARKING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before CarParking_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to CarParking_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CarParking

% Last Modified by GUIDE v2.5 05-Jan-2020 21:17:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CarParking_OpeningFcn, ...
                   'gui_OutputFcn',  @CarParking_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before CarParking is made visible.
function CarParking_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CarParking (see VARARGIN)

% Choose default command line output for CarParking
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes CarParking wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = CarParking_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in togglebutton_connect.
function togglebutton_connect_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton_connect (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton_connect
global arduino

         try
            arduino = serial('COM5','BaudRate',115200);
            fopen(arduino);

         catch e

             arduino = serial('COM5','BaudRate',115200);
            fopen(arduino);
         end
```

```
while get(hObject, 'Value')
    data=fscanf(arduino);
    %disp(isempty(data));
    if isempty(data)
    else
    text=data(1:5);

    if text=="open1"
        set(handles.text_AS1, 'BackgroundColor', [0 1 0] );
        set(handles.text_OS1, 'BackgroundColor', [1 0 0] );

    elseif text=="open2"
        set(handles.text_AS2, 'BackgroundColor', [0 1 0] );
        set(handles.text_OS2, 'BackgroundColor', [1 0 0] );
    elseif text=="open3"
        set(handles.text_AS3, 'BackgroundColor', [0 1 0] );
        set(handles.text_OS3, 'BackgroundColor', [1 0 0] );
    elseif text=="open4"
        set(handles.text_AS4, 'BackgroundColor', [0 1 0] );
        set(handles.text_OS4, 'BackgroundColor', [1 0 0] );
    elseif text=="clos1"
        set(handles.text_AS1, 'BackgroundColor', [1 0 0] );
        set(handles.text_OS1, 'BackgroundColor', [0 1 0] );
    elseif text=="clos2"
        set(handles.text_AS2, 'BackgroundColor', [1 0 0] );
        set(handles.text_OS2, 'BackgroundColor', [0 1 0] );
    elseif text=="clos3"
        set(handles.text_AS3, 'BackgroundColor', [1 0 0] );
        set(handles.text_OS3, 'BackgroundColor', [0 1 0] );
    elseif text=="clos4"
        set(handles.text_AS4, 'BackgroundColor', [1 0 0] );
        set(handles.text_OS4, 'BackgroundColor', [0 1 0] );
    end
    end
    pause(0.5);
end

 fclose(arduino);
 delete(arduino);
```

ARDUINO CODE

//CENTINO, CHUA, NOPIA

```
#include <LiquidCrystal.h>
char mtlb_data; //MATLAB Data
int LED1 = A2;
int IR1 = 2;
int LED2 = A3;
int IR2 = 3;
int LED3 = A4;
int IR3 = 4;
int LED4 = A5;
int IR4 =  5;
int sensorOut[]={HIGH,HIGH,HIGH,HIGH};
int check0 = HIGH;
int check1 = HIGH;
int check2 = HIGH;
int check3 = HIGH;
int currentSlot=1;
int LED;
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

//syntax :- lcd(RS,Enable,D4,D5,D6,D7);
void setup() {
pinMode(LED1,OUTPUT);
pinMode(IR1,INPUT);
pinMode(LED2,OUTPUT);
pinMode(IR2,INPUT);
pinMode(LED3,OUTPUT);
pinMode(IR3,INPUT);
pinMode(LED4,OUTPUT);
pinMode(IR4,INPUT);
lcd.begin(16,2);
lcd.blink();
Serial.begin(115200);
lcd.print("1=open  2=open");
lcd.setCursor(0,1);
lcd.print("3=open  4=open");
digitalWrite(LED1, HIGH);
digitalWrite(LED2, HIGH);
digitalWrite(LED3, HIGH);
digitalWrite(LED4, HIGH);
}
```

```
void ChangeLCD(bool occupied,int currentSlot)
{
  if(occupied==true){
    ChangeLCD(currentSlot);
    lcd.print("close");
    Serial.println("clos"+String(currentSlot));
    }
  else{
    ChangeLCD(currentSlot);
    lcd.print("open ");
    Serial.println("open"+String(currentSlot));
    }

}

void ChangeLCD(int currentSlot)
{
 switch (currentSlot)
 {
  case 1:
          lcd.setCursor(2,0);
          break;
  case 2:
          lcd.setCursor(10,0);
          break;
  case 3:
          lcd.setCursor(2,1);
          break;
  case 4:
          lcd.setCursor(10,1);
          break;
  default:
          break;
 }

}
void loop() {

 switch (currentSlot)
 {
  case 1:
          sensorOut[0] = digitalRead(IR1);
          LED = LED1;
          break;
  case 2:
          sensorOut[1]= digitalRead(IR2);
          LED = LED2;
          break;
  case 3:
          sensorOut[2] = digitalRead(IR3);
          LED = LED3;
          break;
  case 4:
          sensorOut[3] = digitalRead(IR4);
          LED = LED4;
          break;
  default:
          break;
 }

 if(sensorOut[0] != check0
||sensorOut[1]!=check1||sensorOut[2]!=check2||sensorOut[3]!=check3){
  if (sensorOut[currentSlot-1]==LOW)
  {
    digitalWrite(LED, LOW);

      ChangeLCD(true,currentSlot);

  }

  else
  {
    digitalWrite(LED, HIGH);

      ChangeLCD(false,currentSlot);

  }
 }

  if(currentSlot==4){
    currentSlot=1;
    }
    else{
    currentSlot++;
      }
check0=sensorOut[0];
 check1=sensorOut[1];
 check2=sensorOut[2];
 check3=sensorOut[3];
delay(1500);
}
```