TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Addressing the problem

Car parking is a major problem in urban areas in both developed and developing countries. Following the rapid incense of car ownership, many cities are suffering with physical struggle for parking by drivers, wastage of time, congestion and collision with imbalance between parking supply and demand which can be considered the initial reason for metropolis parking problems.

1.2 Why Arduino?

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.

The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board -- you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

Using Arduino uno when can manage the overall functioning of the parking area as it consists of micro-controller which will be operated by the needed function or program.

1.3 Structure of the report

In chapter 2, it provides the overview of the requirements that are needed for the completion of the project and it also includes the hardware requirements and the software requirements. In chapter 3, it provides the information about the implementation of the system that includes the circuit diagram of the hardware components. In chapter 4, it provides the brief information about the program used to control the total working of the project in which some basic functions and header files used in the Arduino IDE is provided. In chapter 5, shows the snapshot of the working model of the project and in chapter 6 and 7 it provides the results founds and the overall conclusions gathered in the project.

2. HARDWARE AND SOFTWARE REQUIREMENTS

2.1 Hardware Requirements

2.1.1 Arduino Uno

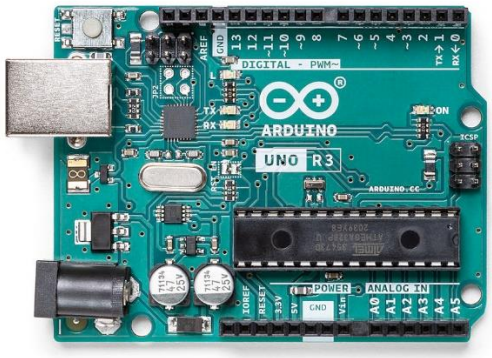


Figure 1- Arduino UNO [1]

Arduino is an open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. Its hardware products are licensed under a CC BY-SA license, while software is licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially from the official website or through authorized distributors.

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards ('shields') or breadboards (for prototyping) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs. The microcontrollers can be programmed using the C and C++ programming languages, using a standard API which is also known as the Arduino Programming Language, inspired by the Processing language and used with a modified version of the Processing IDE. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) and a command line tool developed in Go.

The Arduino project began in 2005 as a tool for students at the Interaction Design Institute Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats and motion detectors.

2.1.2 Breadboard

Figure 2- Breadboard [5]

A Breadboard is simply a board for prototyping or building circuits on. It allows you to place components and connections on the board to make circuits without soldering. The holes in the breadboard take care of your connections by physically holding onto parts or wires where you put them and electrically connecting them inside the board. The ease of use and speed are great for learning and quick prototyping of simple circuits. More complex circuits and high frequency circuits are less suited to breadboarding. Breadboard circuits are also not ideal for long term use like circuits built on perfboard (protoboard) or PCB (printed circuit board), but they also do not have the soldering (protoboard), or design and manufacturing costs (PCBs).

2.1.3 Resistor 220 ohm

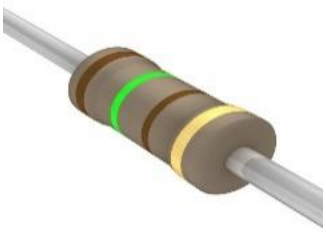


Figure 3- Resister 220ohm [1]

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat may be

used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time, or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in electronic equipment. Practical resistors as discrete components can be composed of various compounds and forms. Resistors are also implemented within integrated circuits.

The electrical function of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than nine orders of magnitude. The nominal value of the resistance falls within the manufacturing tolerance, indicated on the component.

2.1.4 Jumper Wire

Figure 4- Jumper Wire [1]

A jump wire (also known as jumper, jumper wire, DuPont wire) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment. Jumper wires typically come in three versions: male-to-male, male-to-female and female-to-female. The difference between each is in the end point of the wire. Male ends have a pin protruding and can plug into things, while female ends do not and are used to plug things into. Male-to-male jumper wires are the most common and what you likely will use most often. When connecting two ports on a breadboard, a male-to-male wire is what you will need.

2.1.5 LCD 20x4

Figure 5- LCD (20x4) [1]

LCD is a flat display technology, stands for "Liquid Crystal Display," which is generally used in computer monitors, instrument panels, cell phones, digital cameras, TVs, laptops, tablets, and calculators. It is a thin display device that offers support for large resolutions and better picture quality. The older CRT display technology has replaced by LCDs, and new display technologies like OLEDs have started to replace LCDs. An LCD display is most found with Dell laptop computers and is available as an active-matrix, passive-matrix, or dual-scan display. The picture is an example of an LCD computer monitor.

The LCD displays are not only different in terms of heavy than CRT monitors; even the process of working of them is also different. An LCD contains a backlight rather than the firing electrons at a glass screen, which offers light to individual pixels arranged in a rectangular grid. All pixels have a sub-pixel, red, green, and blue, which can be turned on or off. The display appears black if all a pixel's sub-pixels are turned off and appears white if all the sub-pixels are turned on 100%. The millions of colour combinations can be possible with the help of adjusting the individual levels of red, green, and blue light.

2.1.6 I2C Module

Figure 6- I2C Module [1]

I2C (Inter-Integrated Circuit, eye-squared-C), alternatively known as I2C or IIC, is a synchronous, multi-controller/multi-target (master/slave), packet switched, single-ended,

serial communication bus invented in 1982 by Philips Semiconductors. It is widely used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication.

Several competitors, such as Siemens, NEC, Texas Instruments, STMicroelectronics, Motorola,[1] Nordic Semiconductor and Intersil, have introduced compatible I2C products to the market since the mid-1990s.

System Management Bus (SMBus), defined by Intel in 1995, is a subset of I2C, defining a stricter usage. One purpose of SMBus is to promote robustness and interoperability. Accordingly, modern I2C systems incorporate some policies and rules from SMBus, sometimes supporting both I2C and SMBus, requiring only minimal reconfiguration either by commanding or output pin use.

Applications

I2C is appropriate for peripherals where simplicity and low manufacturing cost are more important than speed. Common applications of the I2C bus are:

- Describing connectable devices via small ROM configuration tables to enable plug and play operation, such as in serial presence detect (SPD) EEPROMs on dual in-line memory modules (DIMMs), and Extended Display Identification Data (EDID) for monitors via VGA, DVI and HDMI connectors.
- System management for PC systems via SMBus; SMBus pins are allocated in both conventional PCI and PCI Express connectors.
- Accessing real-time clocks and NVRAM chips that keep user settings.
- Accessing low-speed DACs and ADCs.
- Changing backlight, contrast, hue, colour balance settings etc in monitors (via Display Data Channel).
- Changing sound volume in intelligent speakers.
- Controlling small (e.g. feature phone) LCD or OLED displays.
- Reading hardware monitors and diagnostic sensors, e.g. a fan's speed.
- Turning on and off the power supply of system components.

2.1.7 IR Remote

Figure 7- IR Remote [4]

An infrared (IR) remote control uses light signals sent from a transmitter located at one end of the remote to a receiver in another electronic device.

These devices include televisions, stereos, DVD players, game consoles, and more. The basic operation of an IR remote control is a bulb or set of bulbs at the end of the remote that sends instructions to remote electronic devices using an invisible (infrared) light.

There are different types of IR remotes, from the least expensive with only one IR transmitter to higher-end units that feature several IR transmitters. Electronic devices compatible with IR remotes feature sensors on the front that can detect infrared light and decode the instructions.

2.1.8 IR Sensor

Figure 8- IR Sensor [1]

The IR sensor or infrared sensor is one kind of electronic component, used to detect specific characteristics in its surroundings through emitting or detecting IR radiation. These sensors can also be used to detect or measure the heat of a target and its motion. In many electronic devices, the IR sensor circuit is a very essential module. This kind of sensor is like human's visionary senses to detect obstacles.

The sensor which simply measures IR radiation instead of emitting is called PIR or passive infrared. Generally, in the IR spectrum, the radiation of all the targets radiation and thermal radiation are not visible to the eyes but can be sensed through IR sensors.

In this sensor, an IR LED is used as an emitter whereas the photodiode is used as a detector. Once an infrared light drops on the photodiode, the output voltage & resistance will be changed in proportion to the received IR light magnitude.

2.1.9 LEDs (Light Emitting Diodes)

Figure 9- LED [6]

Light Emitting Diodes (LEDs) are the most widely used semiconductor diodes among all the different types of semiconductor diodes available today. Light emitting diodes emit either visible light or invisible infrared light when forward biased. The LEDs which emit invisible infrared light are used for remote controls.

A light Emitting Diode (LED) is an optical semiconductor device that emits light when voltage is applied. In other words, LED is an optical semiconductor device that converts electrical energy into light energy. A light Emitting Diode (LED) is an optical semiconductor device that emits light when voltage is applied.

When Light Emitting Diode (LED) is forward biased, free electrons in the conduction band recombines with the holes in the valence band and releases energy in the form of light. The process of emitting light in response to the strong electric field or flow of electric current is called electroluminescence.

A normal p-n junction diode allows electric current only in one direction. It allows electric current when forward biased and does not allow electric current when reverse biased. Thus, normal p-n junction diode operates only in forward bias condition.

2.1.10 Micro Servo Motor

Figure 10- Micro Servo motor [1]

The servo motor is an electric motor, which enables continuous determination of precise positions, speeds, and torque via control electronics (servo controller). A software interface with the control electronics also allows precise parameterization and programming for actuation of the motor, which provides a high degree of dynamism and individuality.

Modern servo motors are actuated by a servo controller – both together form the servo drive. With this combination, the motor can be operated with very dynamic movements, high effective torque, and high-power density. Due to the way it works, it is also particularly energy efficient and can achieve high nominal power with relatively small deviation. Costs are therefore saved on multiple levels, which are reflected in better economic efficiency of servo-assisted drive systems.

The greatest strength of the servo motor and the way it works, however, does not emerge until it interacts with the servo controller. Together, software-supported programming is possible, which ensures outstanding precision. Predefined torque, position and speed can be controlled with maximum precision and dynamically adapted to the required movement. Very complex machine processes can therefore be implemented.

2.1.11 Wi-Fi Module

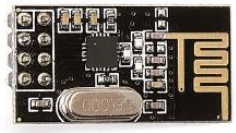


Figure 11- Wi-Fi Module [7]

It is an electronic component on a host device that facilitates access to the internet via a wireless connection. Also, you may refer to them as WLAN modules. In our case, we are interested in the Arduino Wi-Fi Modules.

Hence, an Arduino Wi-Fi module is a WLAN module compatible with Arduino Uno. Thus, they run on Arduino software. The module is useful in the Internet of Things (IoT) applications and mobile devices. It is also popular in wearable electronics, healthcare applications, and home automation projects. Besides, you will also find it in creating applications and smart building technologies.

2.2 Software Requirements

2.2.1 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

Programs written using Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

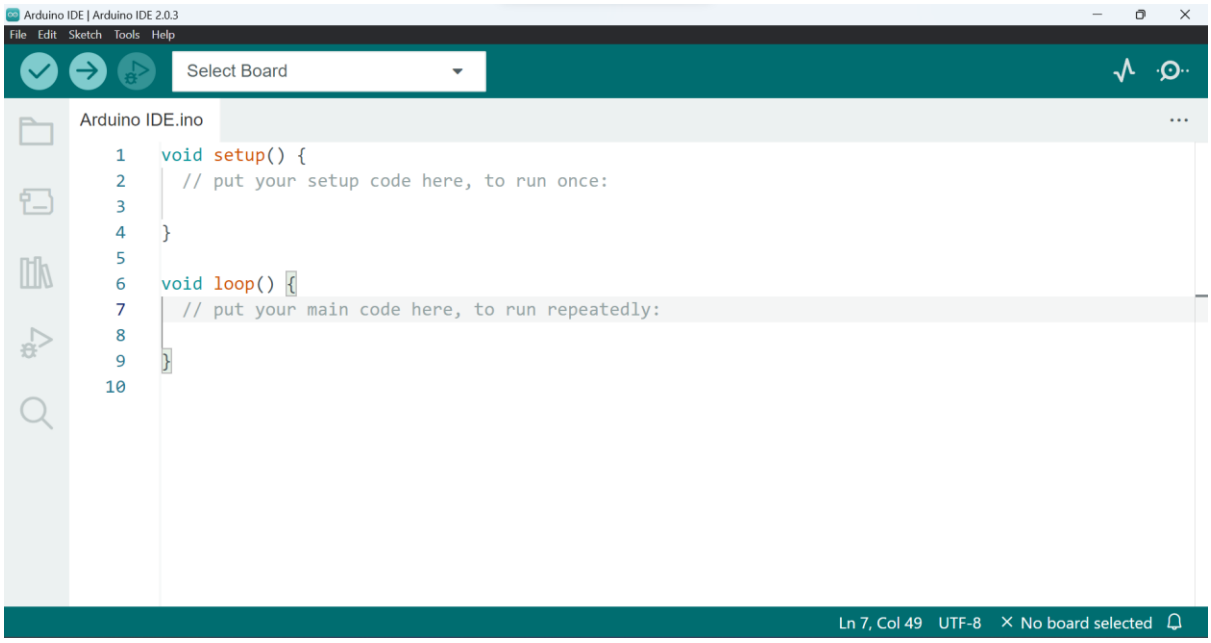


Figure 12- Arduino IDE

2.2.2 Remote XY Mobile Application

RemoteXY is easy way to make and use a mobile graphical user interface for controller boards to control via smartphone or tablet. The system includes:

- Editor of mobile graphical interfaces for controller boards
- Mobile app RemoteXY that allows to connect to the controller and control it via graphical interface.
- The interface structure is stored in the controller.

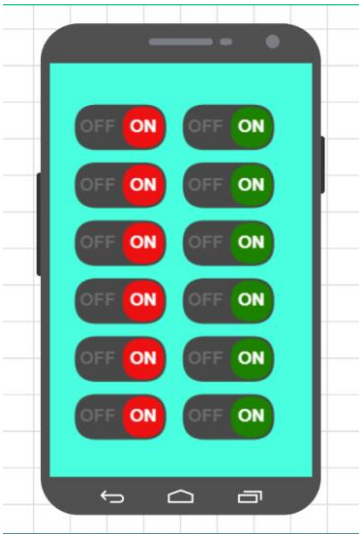


Figure 13- RemoteXY : Arduino control [2]

3. SYSTEM IMPLEMENTATION

3.1 Circuit Diagram

3.1.1 LED

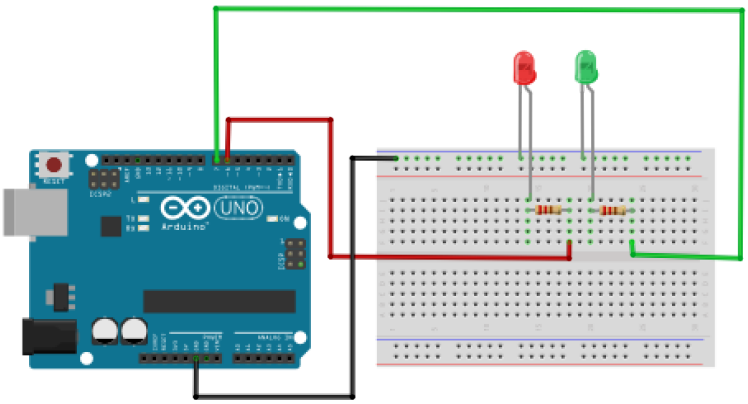


Figure 14-Circuit diagram of LED [3]

3.1.2 Servo Motor

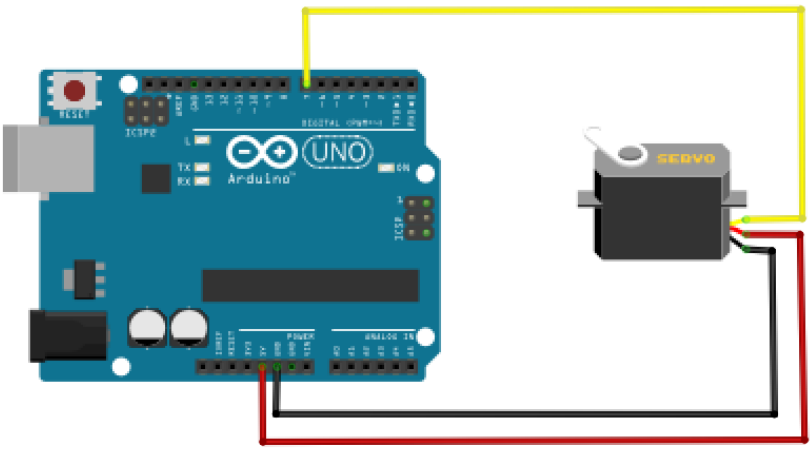


Figure 15-Circuit diagram of servo motor [3]

3.1.3 IR Sensor and Receiver

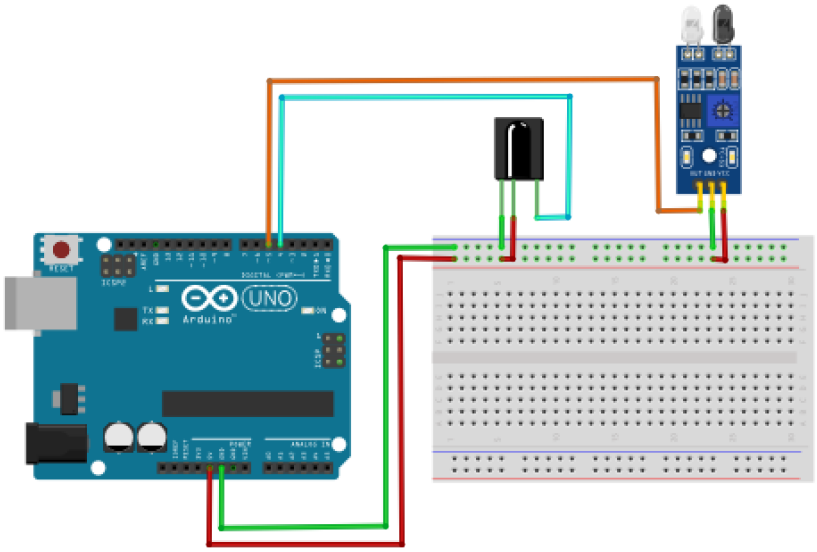


Figure 16-Circuit diagram of IR sensor and receiver [3]

3.1.4 LCD

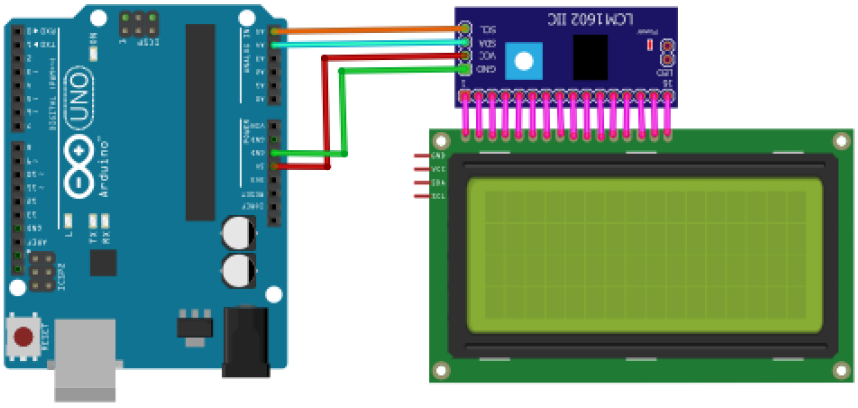


Figure 17-Circuit diagram of LCD [3]

3.1.5 WI-FI Module

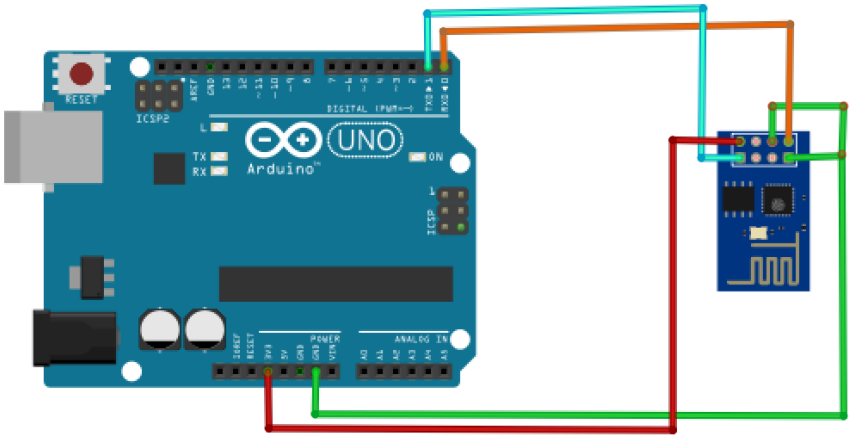


Figure 18-Circuit diagram of WIFI module [3]

3.1.6 Final Circuit Diagram

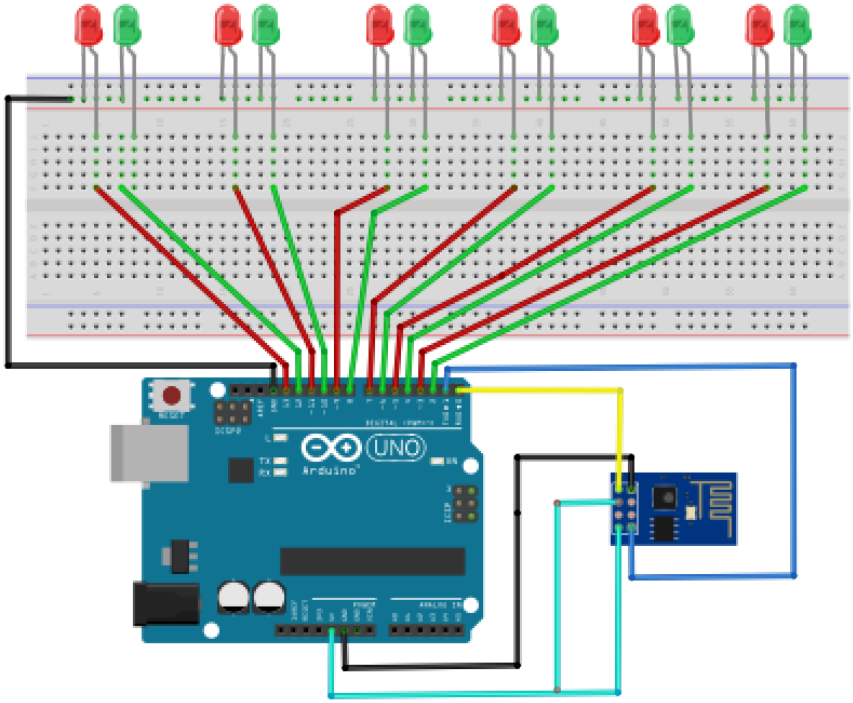


Figure 19-Circuit diagram of LED for slot indication [3]

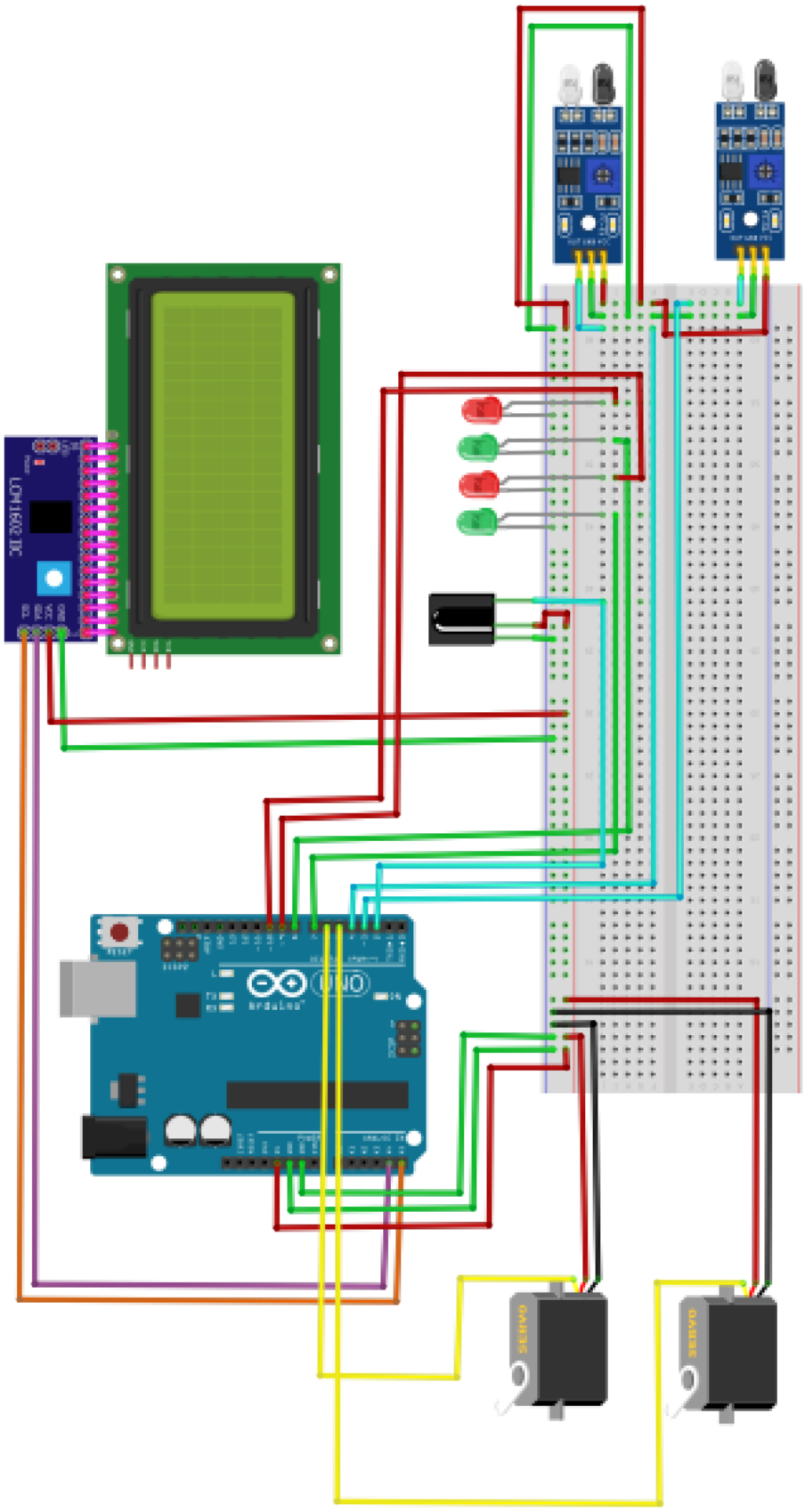


Figure 20-entry and exit gates [3]

4. PROGRAM

4.1 Header files used

#include <Servo.h>

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.
The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

#include <IRremote.h>

Send and receive infrared signals with multiple protocols.

#include <Wire.h>

This library allows you to communicate with I2C/TWI devices. On the Arduino boards with the R3 layout (1.0 pinout), the SDA (data line) and SCL (clock line) are on the pin headers close to the AREF pin. The Arduino Due has two I2C/TWI interfaces SDA1 and SCL1 are near to the AREF pin and the additional one is on pins 20 and 21.

#include <LCD.h>

Defines and Macros for the LCD Controller module.

#include <LiquidCrystal_I2C.h>

This library allows an Arduino/Genuino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs. The library works with in either 4 or 8 bit mode (i.e. using 4 or 8 data lines in addition to the rs, enable, and, optionally, the rw control lines).

#include < RemoteXY.h >

RemoteXY is easy way to make and use a mobile graphical user interface for controller boards to control via smartphone or tablet. The system includes:

- Editor of mobile graphical interfaces for controller boards.
- Mobile app RemoteXY that allows to connect to the controller and control it via graphical interface.

Distinctive features:

- The interface structure is stored in the controller. When connected, there is no interaction with servers to download the interface. The interface structure is downloaded to the mobile application from the controller.
- One mobile application can manage all your devices. The number of devices is not limited.

4.2 Some basic functions used

Setup

Setup is the first function an Arduino program reads, and it runs only once. Its purpose, as hinted in the name, is to set up the Arduino device, assigning values and properties to the board that do not change during its operation. The setup function looks like this:

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

PinMode

The pinMode function configures a specified pin either for input or output: to either receive or send data. The function includes two parameters:

✓ pin: The number of the pin whose mode you want to set

✓ mode: Either INPUT or OUTPUT

In the Blink sketch, after the two lines of comments, you see this line of code:

```
pinMode(led, OUTPUT);
```

DigitalWrite

Within the loop function, you again see curly brackets and two different orange functions: digitalWrite and delay.

First is digitalWrite:

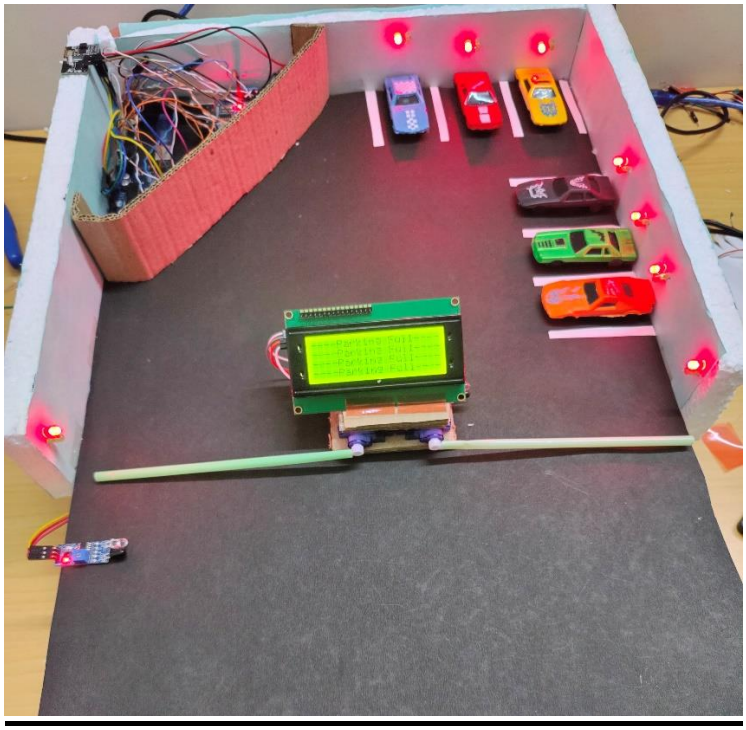
```
  digitalWrite(led, HIGH);   // set the LED on
```

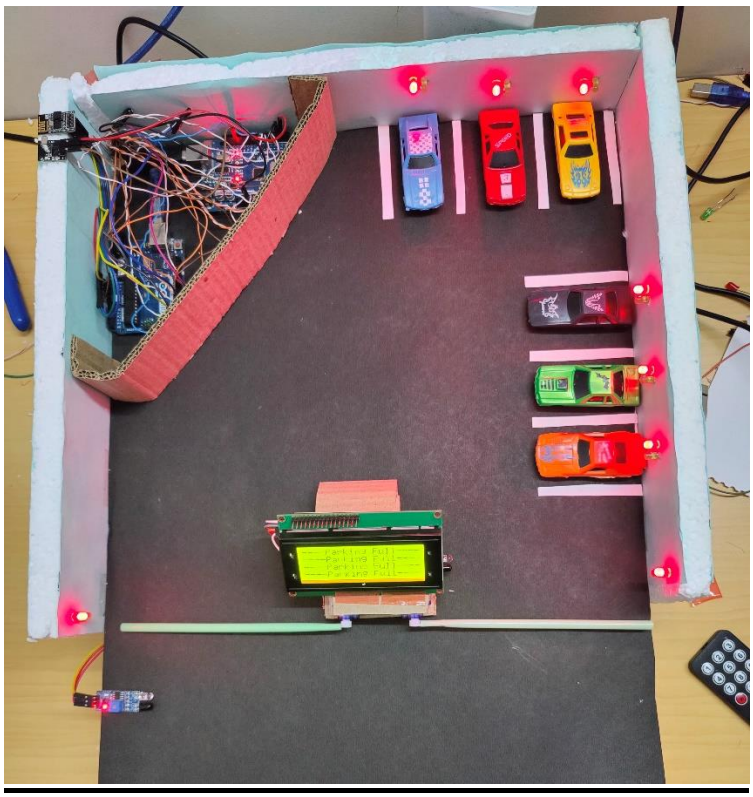
Delay

This function does just what it says: It stops the program for an amount of time in milliseconds. In this case, the value is 1000 milliseconds, which is equal to one second.

```
  delay(1000);          // wait for a second
```

5. Snapshot of the Project





6. RESULTS

Our project consists of 2 Arduino UNO for two purposes. First is for the working of servo motors, IR sensors, LCD for the opening and closing of the gates and lcd for the display of the slots that are empty or occupied in the parking area as well as the total number of slots that are available at the current moment and also if the parking is full it will show as it is full. Second is for the implementation of the slots LEDs that are placed in each and every slots of the parking area so that it can indicates whether the slot is empty or full whenever vehicles arrives or exits as well as it also indicates the place where you are assigned for the parking.

Now, as any vehicle arrives at the entry gate of the parking area it can check the total number of slots available and the slots empty and the slots occupied by the vehicle but as soon as it arrives the range of the IR sensor the sensor sends the signal to the Arduino uno and then it sends the signal to the servo motor to open and along with it a slot will be specified to the one who has to park the vehicle that will be indicated by the LED and the when the vehicle has to go out if the parking area the vehicle will vacate the slot and the slot will be made as green that means the slot is empty and any car can park there and as soon as the vehicle arrives at the exit gate the IR sensor present at the exit gate is going to send the signal to the Arduino and then Arduino will send the signal to the servo motor and the gate will open and as the vehicle leaves the number of occupied slot will be reduced and the number of empty slots will be increased.

Thus the project is helping to manage the parking area efficiently as the congestion or crowd in the parking area is reduced and the person has also the information about the empty as well as the occupied slot.

7. CONCLUSION

Our project helps the drivers to find parking space in unfamiliar city or town or area. The average waiting time of users for parking their vehicles is effectively reduced in this system. The optimal solution is provided by the proposed system, where most of the vehicles find a free parking space successfully. Our preliminary test results show that the performance of the Arduino UNO based system can effectively satisfy the needs and requirements of existing car parking hassles thereby minimizing the time consumed to find vacant parking lot and real time information rendering. This smart parking system provides better performance, low cost and efficient large scale parking system. When car enters the parking area, the driver will park the car in the nearest empty slot when slot is occupied the LED light glows and when slot is empty LED lights are turned off automatically indicating that the parking slot is empty to be occupied. It also eliminates unnecessary travelling of vehicles across the filled parking slots in the area.

REFERENCES

[1] https://www.arduino.cc/

[2] https://remotexy.com/

[3] https://fritzing.org/

[4] https://www.lifewire.com/what-is-an-ir-remote-control-5194485

[5] https://www.amazon.in/s?k=breadboard&crid=1J299W2UP0SVS&sprefix=brea%2Caps%2C529&ref=nb_sb_ss_ts-doa-p_1_4

[6] https://www.bing.com/search?q=LEDs+(Light+Emitting+Diodes)&cvid=e8d17bb2f3324cc0a0b850cfd0896a98&aqs=edge..69i57j0l8.479j0j4&FORM=ANAB01&PC=U531

[7[https://www.ourpcb.com/arduino-wifi-module.html

[8] https://www.theengineeringprojects.com/2019/12/introduction-to-20-x-4-lcd-module.html

ANNEXURE

a. Entry and Exit

```
#include <Servo.h>
#include <IRremote.h>
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

#define I2C_ADDR 0x3F
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

int IRSensor1 = 3;
int IRSensor2 = 4;
int LEDG1 = 7;
int LEDG2 = 8;
int LEDR1 = 9;
int LEDR2 = 10;
Servo s1;
Servo s2;
const int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results results;

float vacant = 6;
float occupied = 0;

void setup() {
  Serial.begin(9600);
  s1.attach(5);
  s2.attach(6);
  pinMode(IRSensor1, INPUT);
  pinMode(IRSensor2, INPUT);

  pinMode(LEDG1, OUTPUT);
  pinMode(LEDG2, OUTPUT);
  pinMode(LEDR1, OUTPUT);
  pinMode(LEDR2, OUTPUT);
  irrecv.enableIRIn();
```

```
  lcd.begin (20,4);
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home ();
}

void loop() {

  int sensorStatus1 = digitalRead(IRSensor1);
  if (sensorStatus1 == 1) {
    digitalWrite(LEDG1, LOW);
    digitalWrite(LEDR1, HIGH);
    s1.write(90);
    delay(1000);
  } else {
    digitalWrite(LEDG1, HIGH);
    digitalWrite(LEDR1, LOW);
    s1.write(0);
    delay(2000);
    occupied = occupied - 1;
    vacant = vacant + 1;
  }

  int sensorStatus2 = digitalRead(IRSensor2);
  if (sensorStatus2 == 1) {
    digitalWrite(LEDG2, LOW);
    digitalWrite(LEDR2, HIGH);
    s2.write(90);
    delay(1000);
  }
  else {
    digitalWrite(LEDG2, HIGH);
    digitalWrite(LEDR2, LOW);
    s2.write(0);
    delay(2000);
    occupied++;
    vacant--;
  }

  if (irrecv.decode(&results)) {
    Serial.println(results.value);
    if (results.value == 16753245) {
      s1.write(0);
      digitalWrite(LEDR1, LOW);
      digitalWrite(LEDG1, HIGH);
      delay(1000);
      occupied = occupied - 1;
      vacant = vacant + 1;
    }
```

```
   else if (results.value == 16769565) {
     s2.write(0);
     digitalWrite(LEDR2, LOW);
     digitalWrite(LEDG2, HIGH);
     delay(1000);
     occupied++;
     vacant--;
   }
   irrecv.resume();
 }

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("    Parking Lot");
 lcd.setCursor(0, 1);
 lcd.print("Total slot   : 6");
 lcd.setCursor(0, 2);
 lcd.print("Vacant slot  : ");
 lcd.print(vacant);
 lcd.setCursor(0, 3);
 lcd.print("Occupied slot: ");
 lcd.print(occupied);

 if (occupied==6){
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("----Parking Full----");
 lcd.setCursor(0, 1);
 lcd.print("----Parking Full----");
 lcd.setCursor(0, 2);
 lcd.print("----Parking Full----");
 lcd.setCursor(0, 3);
 lcd.print("----Parking Full----");
  }
}
```

b. Slots Indication

```
#define REMOTEXY_MODE__ESP8266_HARDSERIAL_POINT

#include <RemoteXY.h>

#define REMOTEXY_SERIAL Serial

#define REMOTEXY_SERIAL_SPEED 9600

#define REMOTEXY_WIFI_SSID "PARKING LED CONTROL"

#define REMOTEXY_WIFI_PASSWORD "happyhappy"

#define REMOTEXY_SERVER_PORT 6377
```

```
#pragma pack(push, 1)
uint8_t RemoteXY_CONF[] =
  {255,12,0,0,0,207,0,16,165,1,2,1,6,10,22,11,1,26,31,31,
  79,78,0,79,70,70,0,2,1,6,24,22,11,1,26,31,31,79,78,0,
  79,70,70,0,2,1,6,38,22,11,1,26,31,31,79,78,0,79,70,70,
  0,2,1,6,52,22,11,1,26,31,31,79,78,0,79,70,70,0,2,1,
  6,66,22,11,1,26,31,31,79,78,0,79,70,70,0,2,1,6,80,22,
  11,1,26,31,31,79,78,0,79,70,70,0,2,1,32,10,22,11,12,26,
  31,31,79,78,0,79,70,70,0,2,1,32,24,22,11,12,26,31,31,79,
  78,0,79,70,70,0,2,1,32,38,22,11,12,26,31,31,79,78,0,79,
  70,70,0,2,1,32,52,22,11,12,26,31,31,79,78,0,79,70,70,0,
  2,1,32,66,22,11,12,26,31,31,79,78,0,79,70,70,0,2,1,32,
  80,22,11,12,26,31,31,79,78,0,79,70,70,0 };

struct {
  uint8_t switch_1;
  uint8_t switch_2;
  uint8_t switch_3;
  uint8_t switch_4;
  uint8_t switch_5;
  uint8_t switch_6;
  uint8_t switch_7;
  uint8_t switch_8;
  uint8_t switch_9;
  uint8_t switch_10;
  uint8_t switch_11;
  uint8_t switch_12;

  uint8_t connect_flag;
} RemoteXY;
#pragma pack(pop)
```

```
#define PIN_SWITCH_1 2
#define PIN_SWITCH_2 4
#define PIN_SWITCH_3 6
#define PIN_SWITCH_4 8
#define PIN_SWITCH_5 10
#define PIN_SWITCH_6 12
#define PIN_SWITCH_7 3
#define PIN_SWITCH_8 5
#define PIN_SWITCH_9 7
#define PIN_SWITCH_10 9
#define PIN_SWITCH_11 11
#define PIN_SWITCH_12 13

void setup()
{
  RemoteXY_Init ();

  pinMode (PIN_SWITCH_1, OUTPUT);
  pinMode (PIN_SWITCH_2, OUTPUT);
  pinMode (PIN_SWITCH_3, OUTPUT);
  pinMode (PIN_SWITCH_4, OUTPUT);
  pinMode (PIN_SWITCH_5, OUTPUT);
  pinMode (PIN_SWITCH_6, OUTPUT);
  pinMode (PIN_SWITCH_7, OUTPUT);
  pinMode (PIN_SWITCH_8, OUTPUT);
  pinMode (PIN_SWITCH_9, OUTPUT);
  pinMode (PIN_SWITCH_10, OUTPUT);
  pinMode (PIN_SWITCH_11, OUTPUT);
  pinMode (PIN_SWITCH_12, OUTPUT);
}
```

```
void loop()
{
  RemoteXY_Handler ();

  digitalWrite(PIN_SWITCH_1, (RemoteXY.switch_1==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_2, (RemoteXY.switch_2==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_3, (RemoteXY.switch_3==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_4, (RemoteXY.switch_4==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_5, (RemoteXY.switch_5==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_6, (RemoteXY.switch_6==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_7, (RemoteXY.switch_7==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_8, (RemoteXY.switch_8==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_9, (RemoteXY.switch_9==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_10, (RemoteXY.switch_10==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_11, (RemoteXY.switch_11==0)?LOW:HIGH);
  digitalWrite(PIN_SWITCH_12, (RemoteXY.switch_12==0)?LOW:HIGH);
}
```