

Contents

Task Division	Error! Bookmark not defined.
Introduction.....	4
Problem Statement	7
Solution	7
Objective	8
Block Diagrams and System Flow/Implementation	9
Input Interface Design.....	12
Ultrasonic Sensor(HC-SR04).....	12
Comparison of Ultrasonic Sensor with PIR sensor.....	14
Significance of Ultrasonic sensor in the design	15
Testing of Ultrasonic Sensors	16
Test Results	16
Photoelectric IR sensor (FC-51)	17
Significance of Photo electric IR electric sensor	19
Discussion – Input Interface Design	21
Output Interface Design	23
Servo Motor	24
1. Positional Servo Motors	25
2. Continuous Servo Motors	26
Comparisons, Selection and Specifications	27
Testing and Connection of Servo Motor.....	28
Code for Servo Motor – For system	32
LED's and CFL's.....	33
Relay	35

Testing and Connection of CFL Bulb	37
Code of CFL – For the system	39
LCD.....	41
Testing and Connection of LCD	42
Code for LCD – For the system.....	44
Complete Combined Outputs.....	46
Results Discussion on Output Interface Design.....	47
Program Development	56
Microcontroller History	56
Intel Microcontrollers	57
Arduino	58
ARDUINO C GRAMMAR/LANGUAGE.....	59
IDE.....	59
ARDUINO STRUCTURE.....	61
DEVELOPMENT	64
Logic	65
Flow chart	66
Pseudocode	69
ULTRASONIC PING SENSOR.....	69
PHOTOELECTRIC IR SENSOR.....	72
CODE & ALGORITHM EXPLAINATION	73
ULTRASONIC PING SENSORS	73
PHOTOELECTRIC IR SENSOR FOR PARKING AID	81
Discussion- Program Development	84
System Integration	88

Circuit Diagram	88
Innovation and Enhancement.....	92
References	94

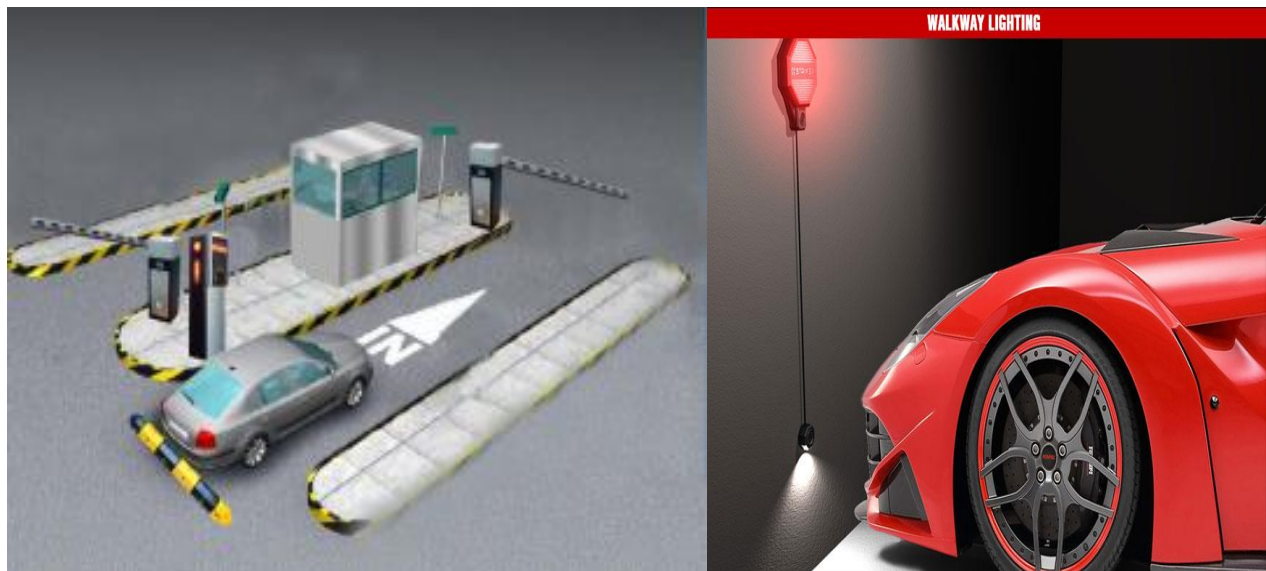
Introduction

The biggest problems to be solved in a city regarding parking space availability are the timely and accurate real-time information of the drivers, the effective management & monitoring of parking spaces to increase revenue for the municipality and the cost-effective law enforcement. Recent studies indicate that the time it takes a driver to find a parking space increases the total time within the vehicle by 24%. Also, transportation studies indicate that 30% of all traffic in the average city center is searching for an available parking space.

Fishing for a parking space is a routine activity and it is estimated that nearly 30% of urban congestion is created by drivers cruising for parking space, according to ITS America's Market Analysis. Also resulting in oil wastage, almost one million barrels of world's oil every day. This ever-growing traffic congestion and uncertainty in the parking availability and payment have thus enforced the need for a Smart Parking system. A Smart parking technology that will help optimize parking space usage, improve the efficiency of the parking operations and help smoother traffic flow.

Smart Parking solutions are designed to provide drivers an ultimate solution on their journey from the beginning to end without searching for parking, cost, travel time etc.

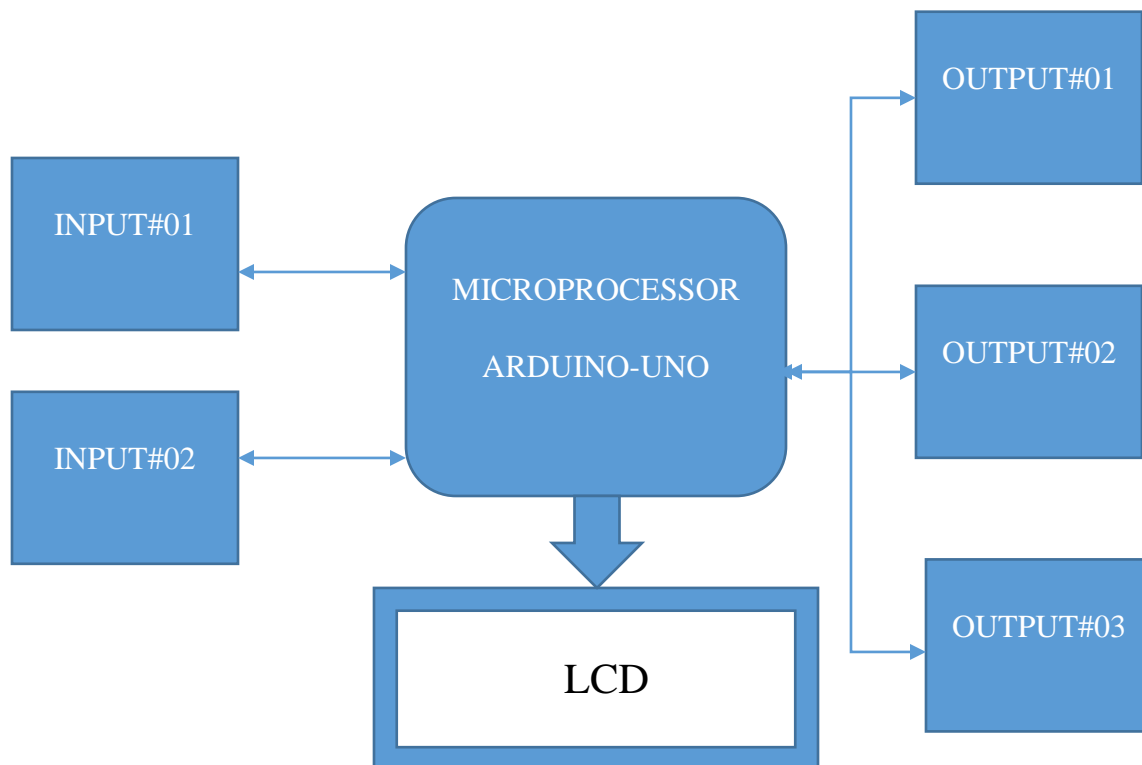
In this assignment we have presented a solution for the problems stated above, the solution is



based

Figure 1

on adequate selection of inputs(sensors) and outputs(light, sound, display etc) corresponding to the main idea of the project while keeping few considerations for example cost, efficiency, product life span, accuracy etc. In this Smart Parking System, we have two inputs and 3 outputs, while LCD being a compulsory component of the project which is not being counted as an output.



During the process of designing the project, many sensors were considered and were tested and all the results were compared in order to select the best components for the project. Since the microprocessor being used in this project is ARDUINO-UNO, which is a microcontroller board based on the ATmega328. Arduino is an open-source, prototyping platform and its simplicity makes it ideal for hobbyists to use as well as professionals. The Arduino Uno has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It simply gets connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started, and the language used for programming is **C-Programming** which is the easiest programming language to operate with.

Problem Statement

Traffic congestions are something which every individual has come across while travelling by road, on top of it finding a parking space and going on for chunk of minutes is the very basic part of a congestion. (Gantelet and Lefauconnier, 2006). In any particular area, street or parking lots there are plenty of vehicles going around for the purpose of finding a parking, this increases the fuel consumption, higher energy wastage and more congestion. (Walker, 2014). According to a study in the U.S the average distance drove to find a parking was a half-mile. Converting that to year it means 47000 gallons of petrol wasted. (Glatz, 2015). Further basic view is that in general a big parking lot or any variable sized parking lot gets too complicated to find parking in and it takes a whole lot of time for them, also usually few drivers are weak in parking and they need extra help to identify, while parking in reverse or from front, and how to park efficiently. For this reason a microcontroller has to be made with minimal tools and particulars available such as Arduino, led, various sensors, etc. The concept being for a parking lot which has specified no of lots available, the barrier of the parking lot doesn't open if the lots are full and this will not let the car go in, and hence they can reverse/take U-turn and go out. At the same time the programming logic to count the available slots and lift the barrier every time the car is sensed is a vital part.

As discussed there are few situations in which the parking lots which are very confined of spaces and might be very difficult to park in. It sometimes becomes very difficult for the drivers to park in these situations, moreover the rear distance between a wall or any other car parked is invisible. Hence here accidents or misjudgments are prone to occur. Similar to the earlier concept another micro controller with sensors and lights has to be designed and implemented which shall read the distance between the car and the wall or car parked behind so as to alarm the driver for how long he/she can go on and reverse park or front. This indication can be given using visual or audio.

Solution

The planned controller offers a lot of ease for user. By implementing it shall allow the viewer's to know whether that parking slot has any space or not just by viewing the LCD screen, also the barrier won't lift if the space isn't available. This is to help in directing people to whether a

particular parking lot has available parking spaces, which is very important to save wastage of fuel and time in this rapidly progressing world.

Further the second part in which the Light will show up when the distance is very close for any car and wall, it is to help the drivers and aid them in parking as it appears many drivers are weak in parking section and often dents can be seen on the car. This controller action is fit on the wall or at the road end of the parking lot which shall sense the car coming close and give an alarm using light to see from the side mirror of the car. Whereas, other means of alarms can be used if required.

Together the Smart Parking Lot and the Smart Parking shall be referred to as Smart Parking system.

Objective

- To make a controller in order to enable cars to enter and exit a parking lot which works on smart concept.
- To make a controller which is an aid to drivers while parking their car and alarm them with the distance left.
- To learn how to use a microcontroller
- To learn to implement logic programming
- To learn real time application and usefulness of microcontrollers

Block Diagrams and System Flow/Implementation

This section explains the general coverage of the input, output and displays connections. This is a very basic representation which is followed by a flow chart. Since the project is made of two parts and both the parts are not interrelated there shall be two separate block diagrams and flow charts. Below the block diagram of the Smart Parking Lot initially, shows how the input ultrasonic sensors which are placed at entry and exit of the parking lot. These sensors sense the car and send the waves to the controller which in turn computes the logic and send the command to the LCD and the outputs connects.

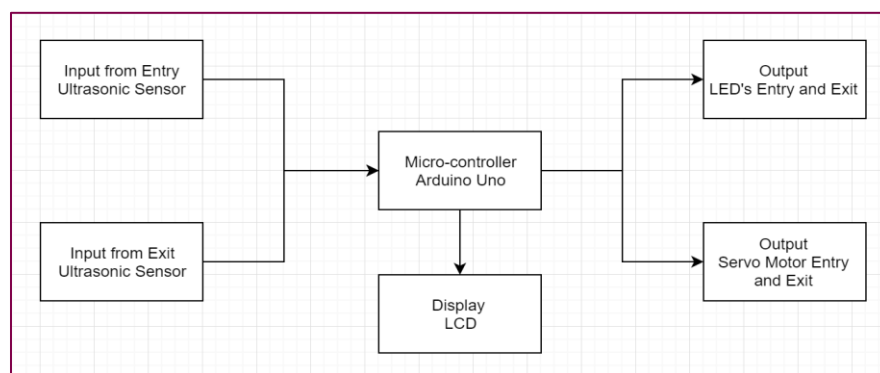


Figure 2 - Block Diagram of smart parking lot

To understand the diagram better the flow chart will be a perfect tool to explain. Below is the flow chart of the smart parking lot. Below the flow chart explains the key features of the smart parking lot, initially when the car is sensed by the ultrasonic sensors at entry or exit point, which come in the sensor's range less than 15mm, these sensors then send the command to the logical unit to check whether the condition $0 < x < 6$ is verified or not. By this it refers to the entry point when the car is sensed, the number of slots have to be checked by the counter whether they are less than 6 or not, if they are then the barrier opens and car goes in and the CFL bulb light comes on to denote that the car can go in. Whereas if the slots are 6 already that denotes the parking as full. This means the car cannot pass in and barrier doesn't open. After this once the car moves in then the sensor again senses the distance to be more than 15 and then closes the barrier and switch off the light.

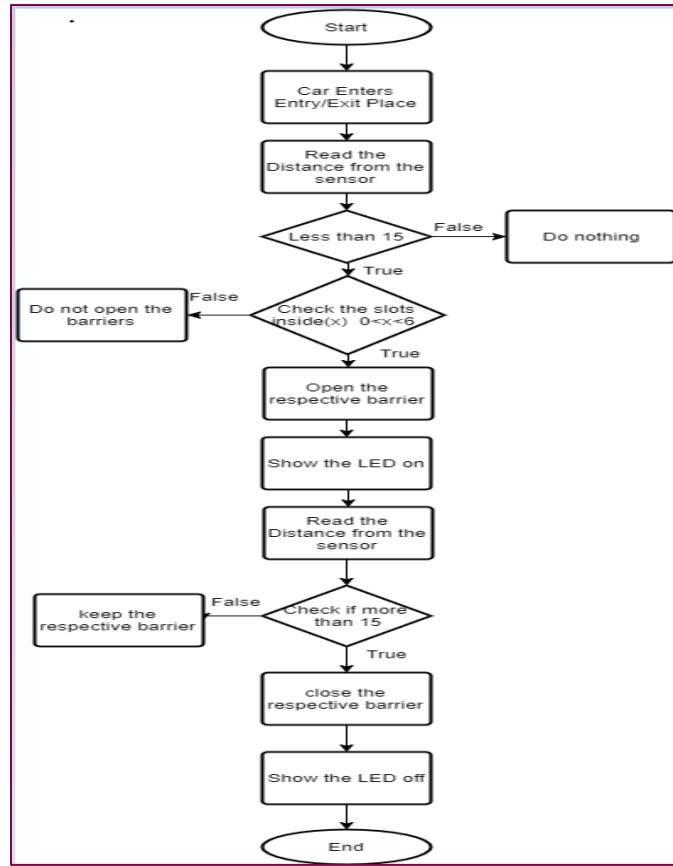


Figure 3 - Flow chart of Smart parking Lot

Further moving on to the Smart parking which takes in one input from the Photoelectric IR. The sensor is kept to sense the distance of the car from the car behind or the wall or any other component where it is fixed. The output of CFL Bulb shows that the car has to stop moving behind once it's on.

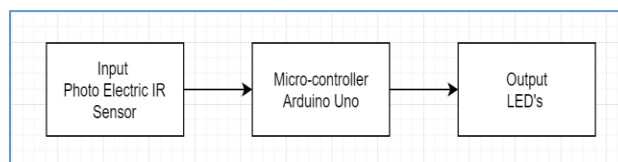


Figure 4 - Block Diagram of Smart Parking

The flow chart of the smart parking can be described as the car when parking in a slot if sensed by the sensor which is attached to the wall, then the CFL bulb will light up indicating the driver that there is no further room. However, once the car is parking within safety distance the bulb will turn off. Below the Flowchart can be seen in the figure.

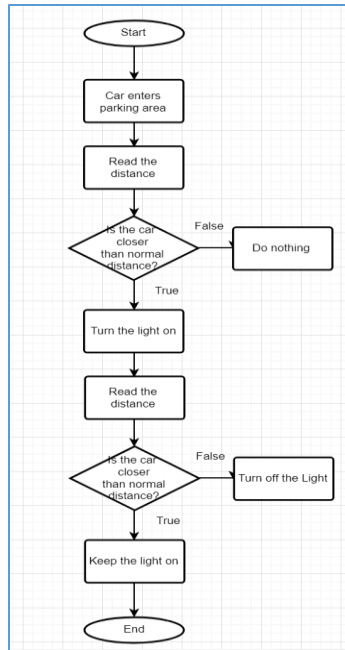


Figure 5 - Flow chart of smart parking

Input Interface Design

For this project 2 inputs are being used

Ultrasonic Sensor(HC-SR04)

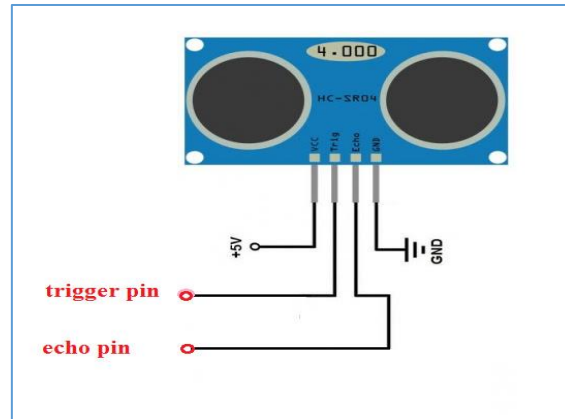


Figure 6

HC-SR04 ultrasonic module can offer non-contact distance whose range sensing is up to 2mm function with the range of 2cm - 400cm. And the range accuracy is up to 2mm. This module includes ultrasonic transmitter, receiver and control circuit. The Vcc and Gnd pin supplies power to the sensor for operation and the trigger sensor is used to trigger the sensor to send sound pulses of high frequency that bounces back of objects. The echo pin is attached to an pin set as INPUT because it reads the time take for the sound pulse signal to go back and forth after it is triggered and bounced. (HC-SR04, 2015)

Module Technical Parameters:

1. Working Voltage : 5V(DC)
2. Static current: Less than 2mA.
3. Output signal: Electric frequency signal, high level 5V, low level 0V.
4. Sensor angle: Not more than 15 degrees.
5. Detection distance: 2cm-450cm.
6. High precision: Up to 0.3cm
7. Input trigger signal: 10us TTL impulse
8. Echo signal : output TTL PWL signal

Mode of Connection:

1. VCC
2. Trig(T)
3. Echo(R)
4. GND

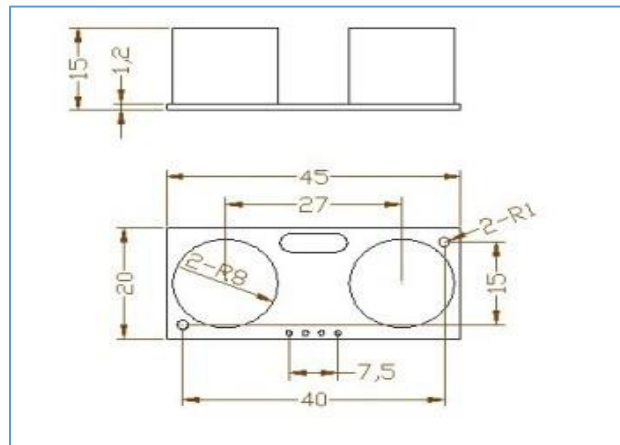
Specification:

Figure 7

Comparison of Ultrasonic Sensor with PIR sensor

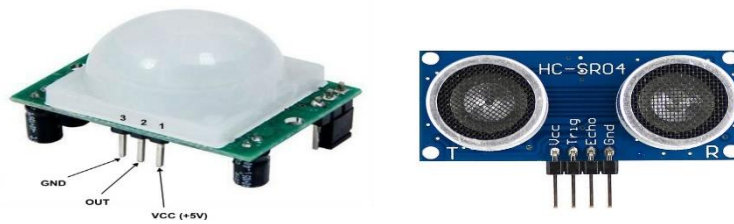


Figure 8

PIR sensors detect occupants' presence by sensing the difference between heat emitted by moving people and background heat. Ultrasonic sensors detect the presence of people by sending out ultrasonic sound waves into a space and measuring the speed at which they return. They look for frequency changes caused by a moving person.

PIR sensors require a direct line of sight between the sensor and occupants in a space. Because of this requirement, managers can strictly define the sensor's coverage. Ultrasonic sensors cover the entire space and do not need a line of sight. As a result, they can detect people behind obstacles. They also are more sensitive to minor motions, such as hand movements.

COMPARISON	
COST	Ultrasonic is cheaper than PIR sensor
COMPLEXITY	PIR sensors are more complicated than the other sensors because there are multiple variables that can affect the sensors input and output.
CALIBRATION TIME	Long calibration time taken by PIR sensor than ultrasonic
SENSITIVITY	PIR sensors are insensitive to very slow motions or the object (i.e. a body) in standing mode while ultrasonic works fine in this condition

Significance of Ultrasonic sensor in the design

After all the comparisons were made between PIR sensor and Ultrasonic sensor, the most reliable and suitable input sensor was selected which is the ultrasonic sensor(HC-SR04). In our design, we have two ultrasonic sensors, one is responsible to detect the car at the entrance point followed by the lifting of the barrier making way for the driver and the second ultrasonic sensor was placed at the exit point of the parking system and the same process of making way takes place. Bulk capacitor is also used for decoupling purposes.

Since the barrier is connected to the servo motor which triggers only when the ultrasonic sensor detects a car it triggers the servo motor and the motor rotates having a delay of 5 seconds which means the barrier lifts for 5 seconds. And at the same time as soon as the barrier returns to its original position the count of parking slots decreases implying number of slots remaining which is being displayed on LCD. In our case we have fixed 6 parking slots so our count starts from 6 slots. Coming to the exit point, the exit sensor will trigger the servo motor again with the same procedure which being followed the entrance point, but in this scenario the number of slots increases as the barrier comes back to its original position. Limitations has been programmed for the number of slots which is 6 and 0, if the number of slots are full and not vacant the barrier for entrance will not lift up even if the ultrasonic is detecting a car but due to the limitation the car will not be able to

enter in the parking

lot.

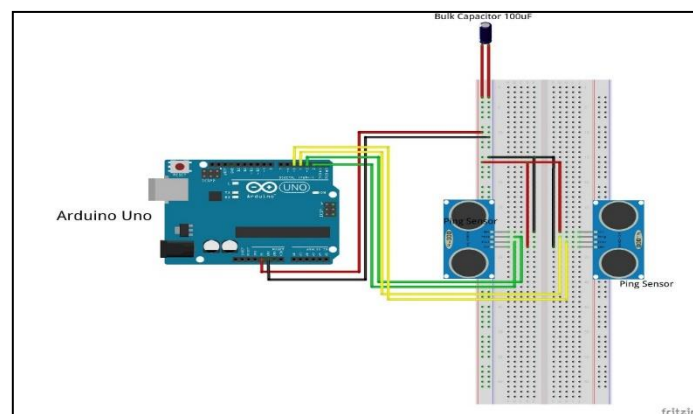


Figure 9

Testing of Ultrasonic Sensors

Test Results

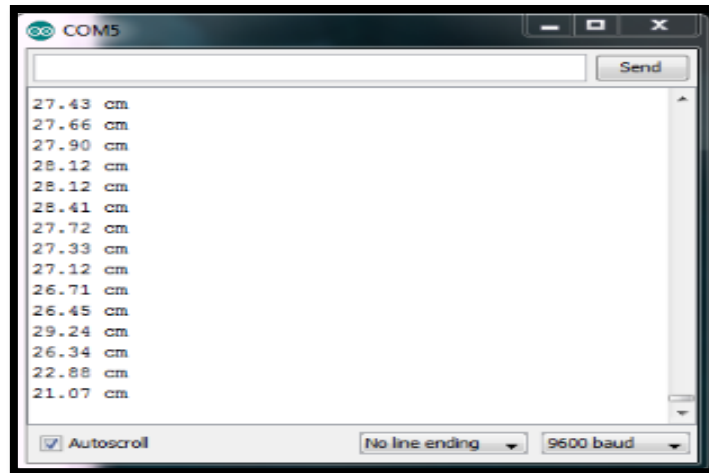


Figure 10

Referring to the above figure, a serial monitor can be seen having values which are fluctuation since the testing was being done of ultrasonic. An object was moved in a back and forth manner in front of ultrasonic, the value 21cm indicates the closeness of the object to ultrasonic and 29.24cm being the farthest distance in between the object and ultrasonic. After this testing was done then a specific value of 15cm was selected for the application in-order to trigger the servo motor, for example if the object was 15cm or was closer to the sensor the servo will be triggered and the barrier will be lifted.

Photoelectric IR sensor (FC-51)

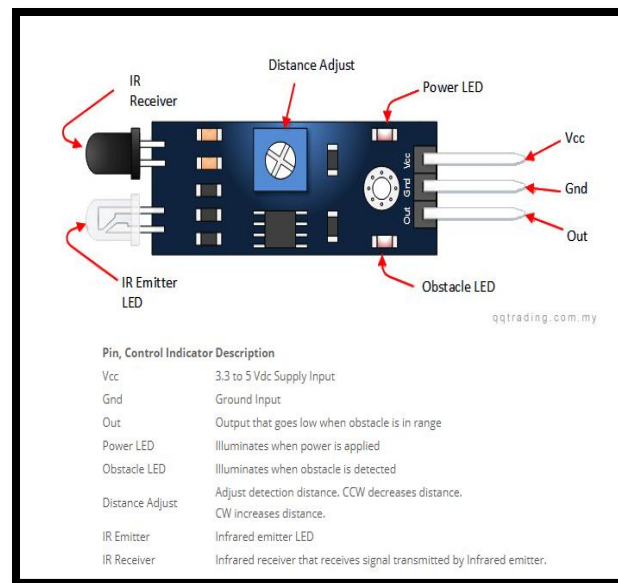


Figure 11

Module Description:

The sensor module light is adaptable to the environment, it has a pair of infrared transmitting and receiving tube, tube infrared emit a certain frequency, when detecting direction meet with obstacles (reflecting surface), reflected infrared receiving tube, after the comparator circuit processing, green indicator will light up, at the same time signal output interface to output digital signal (a low level signal), can be through the potentiometer knob adjust the detection distance, effective distance range 2 ~ 30cm, working voltage is 3.3V to 5V. The sensor detection range can be through the potentiometer to adjust and have small interference, easy to assemble, easy to use, etc, can be widely used in robot obstacle avoidance, obstacle avoidance car, line count, and black and white line tracking and so on many occasions.

Module Parameters:

1. When the module detects obstacles in front of the signal, the green indicator on the circuit board light level and at the same time the OUT port output low level signal, the detection module

from 2 ~ 30cm, 35° detection Angle, test distance can be adjusted through the potentiometer, adjustable potentiometer clockwise, the detection distance increases; Counter-clockwise tuning potentiometer, detection distance is reduced.

2. sensors, active infrared reflection detection, therefore the reflectivity and shape of the target is the key of the detection range. White one black detection range, minimum maximum; Small area of the object distance, large distance.

3. the sensor module output port OUT can be directly connected to the microcontroller IO port can, also can drive a 5v relay directly; Connection mode: the VCC - VCC; GND - GND; The OUT - the IO

4. the comparator USES the LM393, working stability;

5. can be used for 3 to 5v dc power to power supply module. When power on, the red power indicator light lit;

6. 3 mm screw holes, easy fixed, installation;

7. circuit board size: 3.2CM*1.4CM

8. each module is shipped already compare threshold voltage through the potentiometer to adjust good, not special circumstances, please do not arbitrarily adjust the potentiometer.

Module Interface Specification:

- VCC voltage is 3.3V to 5V converter (which can be directly connected to 5V single-chip microcontroller and 3.3V)
- 2. GND external GND
- 3. OUT of small plate digital output interface (0 and 1)

Significance of Photo electric IR electric sensor

Photoelectric IR sensor was selected over LDR because LDR is highly inaccurate with a response time of about tens or hundreds of milliseconds. Since this sensor was being used for parking aid, that is why the sensor has to be accurate with its range and reaction so that this technology can assist the driver in keeping safety distance from the wall.

As it can be distinguished from figure 1 and figure 2, the difference is the reaction of the output based on distance. As car comes closer to the wall the danger LED lights up indicating the driver to stop the car at that moment to avoid the accident.

In figure number 3 serial monitor is displayed having values from the range of 153 to 885. So 500 was kept as the fixed distance which is the safety distance in this project for parking aid.

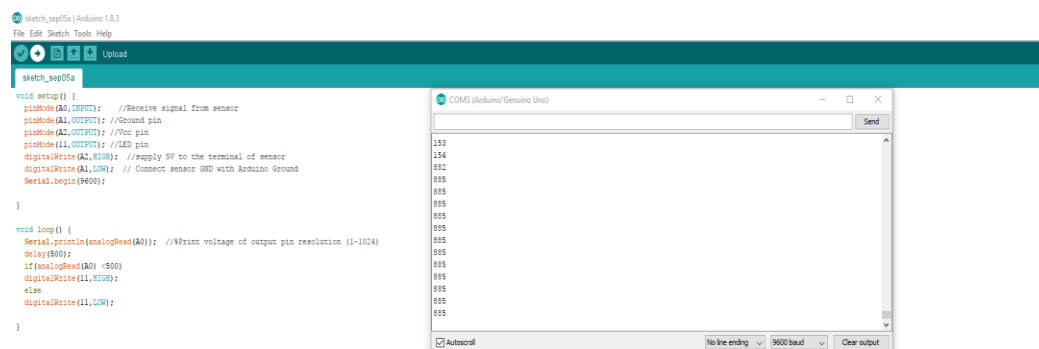
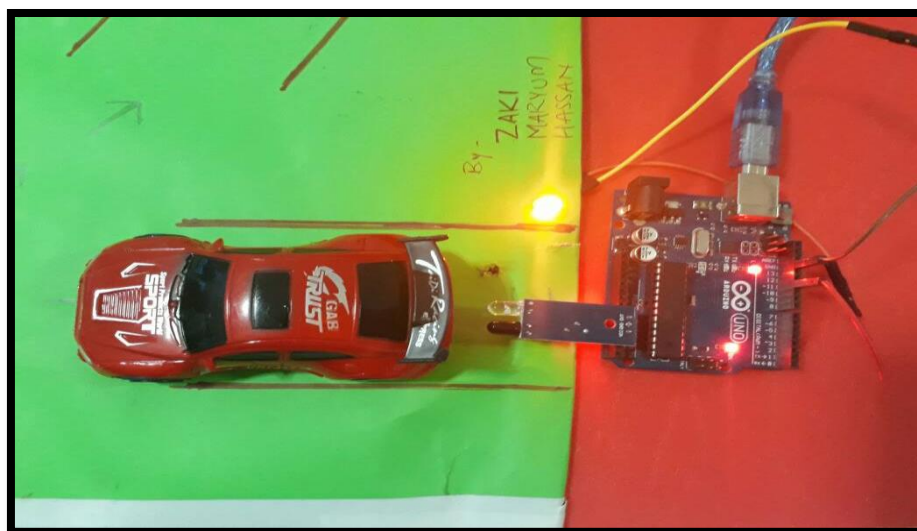
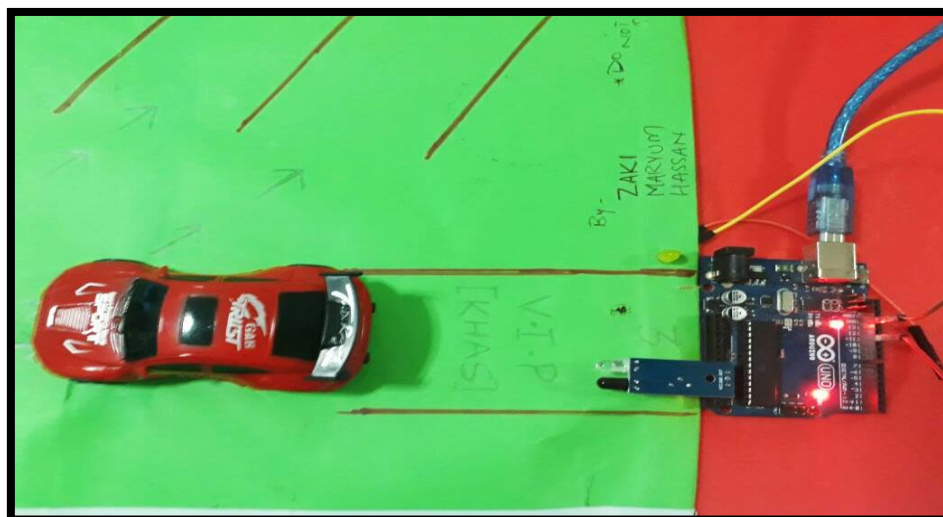


Figure 14



Discussion – Input Interface Design

Designing a smart parking system was a challenging task, few problems were faced among them one was the selection of inputs for the smart parking system. Comparisons were made between sensors so that a suitable sensor could be selected. While comparing few considerations were kept in mind which were the cost of the sensor, durability, accuracy and connecting complexity. For Input, two ‘Ultrasonic sensors’ and one ‘Photoelectric IR sensor’ was used. For both the inputs firstly range was observed and monitored and then fixed range with specific value was selected according to the application.

Another problem faced was getting incorrect output from the ultrasonic sensor even sometimes the values were in negative when observed in serial monitor because Sensors can include digital circuitry which can generate high frequency content, thanks to square waves. You cannot give these sensors low impedance power paths because of the already long wires; However, you can limit the high frequency content coming from these sensors to your board. To solve this problem bulk capacitor was used of 10 μ F for decoupling. After which the results attained were satisfactory.

One of the other problem was the unit of distance, the value which was being displayed on the serial monitor while testing ultrasonic did not have any indication of unit for example cm or mm so to solve this problem we had to look for a data sheet which had ample information about the unit and measurement criteria of a particular sensor on which researching we got to know it was centimeter and then we proceeded with limiting the range of the sensor according to our application.

Photoelectric IR sensor’s information and connection was very difficult to find because the information available on internet is very rare to find which resulted in creating difficulties for us to calibrate with this sensor. Then calibration was done using the technique stated below.

1. Present the sensor with the darkest shape at the smallest distance you need to detect.
2. Turn the pot until the sensor trips.
3. Turn the pot back just slightly until the trip clears.

Since 3 inputs and 3 outputs with a LCD display were connected in the system so the wiring was an issue as well since all sensors has to be grounded and must be supplied with VCC so it became quite messy and earlier wires of different colors were used for grounding and source later on to reduce the confusion all 3 inputs were grounded using the bread board and to differentiate for each input or output corners of breadboard were used having rails A and B, one having source other having the ground connection. And then accordingly the wires were connected corresponding to the placement of sensors.

Output Interface Design

For any system as it is known consists of and input, output, the process and sometime a feedback. Input is considered what is fed into the system whereas the output is the result of the input fed after being processed according to the input data. In literal terms, input is the desired output. A simple example of a process which contains both output and input is the recording of a sound and then playing it back, here the voice recorded is the input which is then processed and reconstructed to play it back.

In a system or a controller which is to be designed there are specific output key terms and points to be noted. Firstly to know what kind of inputs are needed, then what is the output response of the system and what is expected, thirdly what process is needed to be done in order to get the output, fourth and the final one what data has to be kept intact so as to keep the output result in check. (Study.com, n.d.)

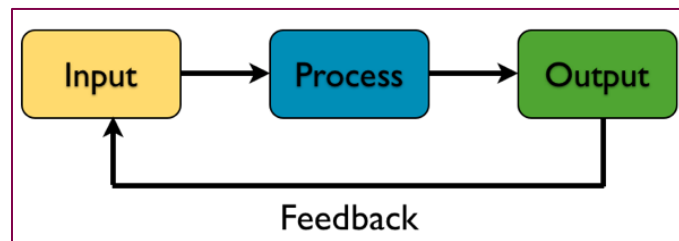


Figure 15 – Block of a system

There are various ways to represent the output of a system, they can be via text, graphics, audio, video and so on. Hence the term output design focuses on type of outputs needed, the controls to manage the output (wiring, connections, drivers, switches), etc. It is highly vital that the output chosen satisfy the user requirement and serve the purpose of the system. (www.tutorialspoint.com, n.d.)

Basic terminologies which are defined have to relate to the Smart parking systems outputs. The system uses a microcontroller, arduino uno, which in turn has its own output pins given and most pins on it are set digital. These pins are to be connected to all the external outputs used and the pins give out the wave as defined in the program. These output pins can have a high or a low as set by the program for the required situation, thus to control external outputs these ports must be connected to it. These external outputs take supply from the control and the signal how to work,

in other cases where the larger outputs are connected they need to be supplied externally. (W9xt.com, n.d.)

With relation to the system designed there were different outputs used. Firstly for the smart parking lot there was a necessity to show the lots available, this can be done using text output, secondly after the computations the car which is sensed has to be sent in or rejected thus it had to be shown using a gate for entry or exit, finally the signal has to be passed to the car to move forward and go in using a visual light as rules of the road are as such.

Next, for the smart parking lot the distance has to be warned to the driver and similar fashion text display has to be used, further the alarm has to be put on with visual and audio to assist better.

These determined outputs were put in to consideration and relative devices/materials were selected. To understand better the following sections will explain the details of each respectively.

Servo Motor

As discussed earlier the car when sensed by the sensors need to pass in to the parking lot which are controlled by the barriers, these barriers are controlled by motors on both the sides.

The reason why a servo motor is taken into account is the basic principle of the servo motor to have a feedback system, which gives the information about the shaft of the motor to the controller. This prevents any error which might occur while using a stepper motor which takes the move command only. (Robotiksistem.com, n.d.)

The different types of servo motors are the positional rotation servo, continuous rotation servo, linear servo. As the name suggests the positional rotation servo goes up to 180 degrees, whereas the continuous one keeps rotating as set, linear servos are similar to rotation but with more mechanisms. (Science Buddies, n.d.).

The basic working of a servo motor functionality looks like this:

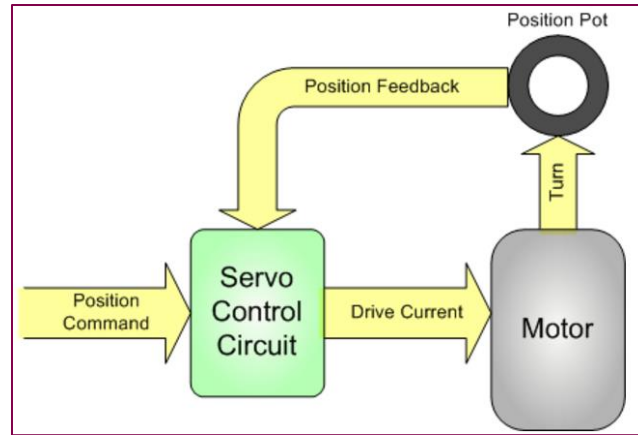


Figure 16 - Working of servo – Source (Earl, 2015)

Since the Linear servo are difficult to find and very less in number also it is used for larger applications, the main focus is on the Positional Rotation servo and the Continuous Rotation servo. (Coker, 2016). The theory behind them and the concept and comparison are as follows.

1. Positional Servo Motors

These are the most widely used motors for accurate and narrow error allowed for positioning. RC motors can go up to a rotation of 180 degrees, containing a DC motor inside, controller part, a gear unit, and a position and speed sensing device typically potentiometer. Different types in terms of sizes are available such as mini, micro, etc. They are mostly low cost and can easily be configured by microcontrollers for usage accordingly. Application as such used in Robotics industry, Boats, aircrafts, etc. (Earl, 2015)

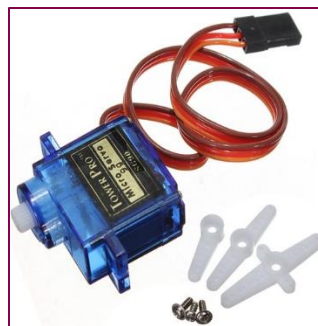


Figure 17 - SG90 servo

They come with three wires, one for the voltage supply, one for ground and one for the command. The advantages of this motor include low cost, wide ranges available in different sizes

and shapes with ranges of torques, and controlling ease. On the other hand the drawbacks include that the rotation is only till 180 degrees.

Specifically RC Micro SG90 in the positional rotation motors, is used with the microcontrollers for projects demonstrations, while to have a real time application a relatively larger motor with external supply and relay has to be used. RC Micro SG90 has a duty cycle of 1-2ms, PWM of 20ms (50Hz) and of 5V power and signal. (Datasheet). PWM is the square wave generated by the microprocessors which is turned on and off within the given limits.

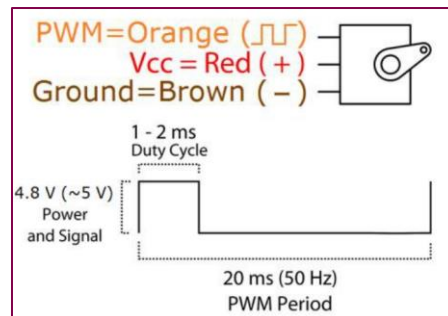


Figure 18 - Pin Diagram

The orange wire which gets the signal is the one which controls the motors shaft, once up to the maximum position, the motors potentiometer will end the supply and wait for the signal again to put it down.

2. Continuous Servo Motors

The shaft of this motor can rotate in any direction and with variable speed. Further the shaft can also rotate for any given duration of time without any stopping. The signal given in to the control unit takes the information about the speed, duration, direction, etc. They are used in robot driven industries and fields.



Figure 19 - Continuous Servo

Some advantages include being low cost, compact, and again simple to control, drawbacks include that they can be only used for very small loads, and the position to stop the shaft is not defined. (Electronics Hub, 2016)

Comparisons, Selection and Specifications

The comparison of the previously described motors is to see how and where they fit in terms of using with the microcontroller.

Positional	Continuous
Low Cost – up to RM10	Low Cost - up to RM10
Rotates 180 degrees or less	Rotates infinitely if set.
Variable sizes	Variable sizes
Adjustable to different loads	Less heavy loads only

From the table above we can see that there is not much difference between the two motors, hence it is vital to see which one suits best to the microcontroller application of the smart parking system. The application requires a motor which can lift up when sensed the car and get back to its rest position after the car is gone. In a typical real time application we can go with the positional servo since it will limit the barrier to stop at a certain angle and not keep rotating continuously. Hence the chosen would be the SG90 RC servo motor which can be connected to the micro controller. The specifications from the datasheet can be seen in the table below.

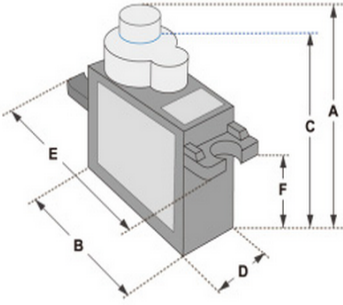
	Dimensions & Specifications
	A (mm) : 32
	B (mm) : 23
	C (mm) : 28.5
	D (mm) : 12
	E (mm) : 32
	F (mm) : 19.5
	Speed (sec) : 0.1
	Torque (kg-cm) : 2.5
	Weight (g) : 14.7
	Voltage : 4.8 - 6

Figure 20 – Datasheet Specifications – Source Anon, (n.d.). SG90 9g Micro Servo

For the servo motor to be known by its specifications we need to determine its current flow, current drawn in other words, the voltage required, the resistance, the torque etc. The torque can be determined by the distance (in cm) and the weight on it. The figure shows the torque to be 20kg-cm, below shown an example:

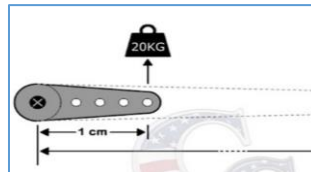


Figure 21 - Torque Determination

Testing and Connection of Servo Motor

Testing is the most essential part of output components. Since there are two servos used for the Smart parking system and both are identical. The circuit is made for both the servos separately to be checked. The servo motor requires 5V to work, and hence an external battery of 9V is added and a regulator which controls the voltage is added to supply proper and adequate voltage to both servo motors. Since there is a bulk capacitor acting as a decoupling capacitor, two capacitors of 0.22uF are used for both the servo motors and two 10uF capacitors are connected to with the voltage regulator, the system integration part executes and explain further on this. The connection looks like below:

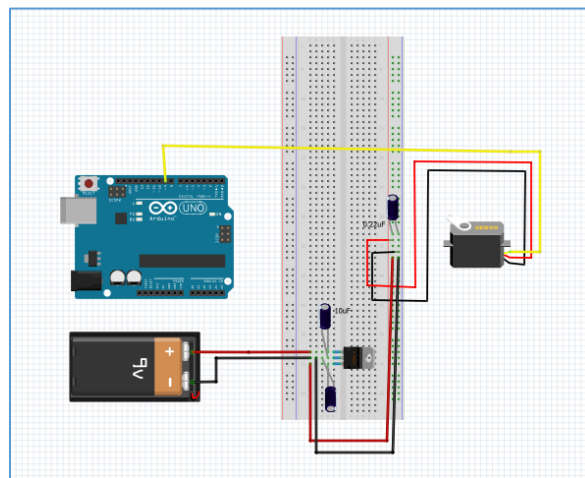


Figure 22

The pin number 9 from the Arduino goes to the signal controller wire of the servo motor for the servo present at the entry point of the parking slot. The additional supply of 9V battery is given to the breadboard. This is then passed through the voltage regulator which cuts the supply to 5V and hence this 5V is then passed to the servo motors showing as a red wire. The task of a voltage regulator is to cut the additional voltage and use and pass on as much as it is intended for, here the voltage regulator is used is LM7805. Which limits the supply to 5V as the servos only need 5V to operate. The capacitors of 10uF are placed near the voltage regulator. A small capacitor of lower capacitance. 22 uF is connected near the load & that is called local capacitor. The reason for powering the servo motor externally is because during the servo motor sweep, it draws a lot of current which exceeds the limit of drawing 10mA from processors and also causes noise in the circuit. Hence, the supply line for the servo motor is separated from other components.

The code for testing is generated from Arduino examples codes, sweep for servo in which the servo is given a particular angle and it keeps going till it reaches that angle and then comes back to the starting position. The delay in the code will generate the PWM sent to servo and hence the duty cycle and such can be generated. The signal pin of the servo as shown in the previous figure must be connected to digital pin 9 and thus should be the declaration in the code. Using this code the servo will turn to 90 degrees and return back to its position with a speed of PWM 20ms. The speed can be varied for different purpose. The code for testing is shown below:

```

/* Sweep
by BARRAGAN <http://barraganstudio.com>
This example code is in the public domain.

modified 8 Nov 2013
by Scott Fitzgerald
http://www.arduino.cc/en/Tutorial/Sweep
*/

#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 90; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position
  }
}

```

Figure 23 - code testing

The connection of servo can be shown on the real circuit as below:

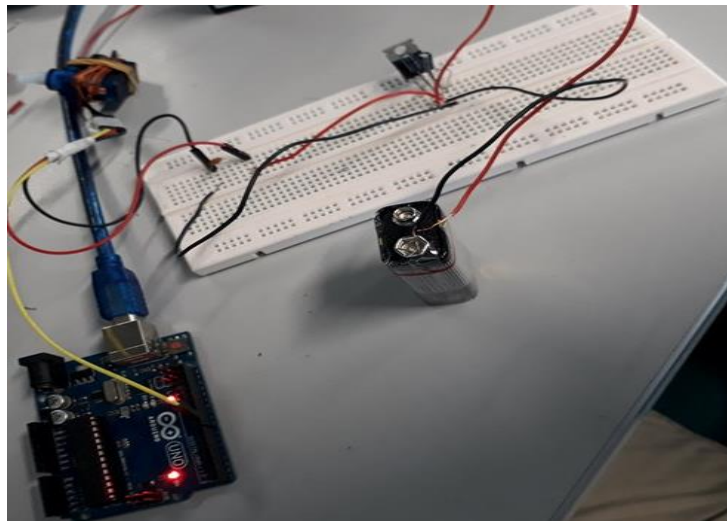


Figure 24 - Connections

To measure the current drawn by the servo motor, an open circuit has to be performed. From the third pin of the voltage regulator has to be pinned elsewhere and then the red probe of the multi-

meter should be in series with the Vcc pin of the servo. At different angles the current drawn can be taken down and noted, namely, the initial start of the motor, the rotation, and the final rest.

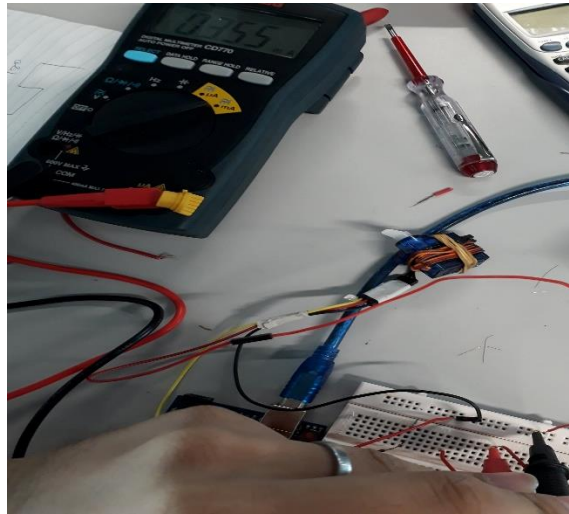


Figure 25 - Testing

The voltage supplied to the servo motor can also be varied to see the current it draws while the motor is stationary. This can be done using a potentiometer.

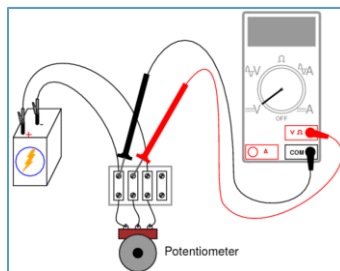


Figure 26

The duty cycle can also generated according to the pulse width given in.

Code for Servo Motor – For system

Now look at the code which is generated for the servo motor operation in the system. The first is the library file which is servo.h, this will bring in all the commands that can be used related to the servo motor.

```
#include <Servo.h>
```

Since we have two servo motors we need to define the Servo with a name of choice.

```
//Defining the servos
```

```
Servo servoIn;
```

```
Servo servoOut;
```

Here the pin number 6 is connected to the servo motor present at the exit side.

```
pinMode(6,OUTPUT);
```

```
servoIn.attach(6);
```

```
servoIn.write(0);
```

Next the pin number 7 is connected to the servo motor present at the entry side.

```
pinMode(7,OUTPUT);
```

```
servoOut.attach(7);
```

```
servoOut.write(0);
```

For the Logical part get the distance from the input sensors and specify the slots are greater than zero, this means it should not go less than zero since the parking lot values can only be a positive values. The servoIn.write(0) opens the servo gate.

```
{ if (..... && slots > 0)
{
    slots = --slots;

    servoIn.write(0); // Open servo gate
```



```
....
```

```
..... }
```

The other part of the logical program is to bring back the servo motor back to position.

```
else if (.....)
```

```
{.....
```

```
servoIn.write(80);
```

```
.....
```

```
}
```

Similarly for the same entry servo motor the code has to be developed. The only difference is the maximum slots have to be given is as 6. That means once the count has reached 6 the servo gate will not open anymore.

```
if (..... && slots < 6)
```

The rest of the codes look same like the exit servo which changed name to servoOut.

LED's and CFL's

Since one of the output has to be kept as the visual light, it had to be a light aid or blinking light. Further due to constraints of a real time application it was necessary to include bulbs, Leds, and CFL's to the comparison and conceptual part. Further in this section we can see that how the selection from these was done.

The tiny Light Emitting diodes are lit up when current passes through them. The internal of an LED is considered as a P-N junction. The energy given out is from the release of photons which is the form of a monochromatic light of nanometer of a single wavelength. The basic advantages include very less power is required, durability is more with good efficiency, and they do not get warm after use, lower cost etc. (WhatIs.com, n.d.). One of the application of LED's is that they are used as indicators in various places.

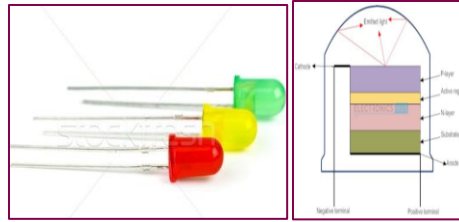


Figure 27 - LEDs

LEDs come in various sizes and shapes, but a drawback is that they are not ideal for illuminating large areas. With a range of RM 15 to RM 20, LED's are more costly than others.

CFL which denotes Compact Fluorescent Light bulbs are a major contributing factor in energy efficient lighting systems. Easily fit into a normal lamp or bulb sockets. They need to pass current through them which contain argon and mercury glass, the fluorescent coating gives the light. A minimum of 11,000 hours are expected from CFLs, and the most convenient. They do not get dim and are not easily and work good in outdoor temperatures. They come in more shapes and sizes than any other type. (Anon, n.d.)



Figure 28 - CFL bulb

As a requirement of taking real time application output it was essential to convert this segment of the output to a real time bulb. Since from the comparison we can see the LED bulb is of higher price and CFL is at a low cost. Moreover, since its just a prototype the durability was not the main concern. However we can also see the size of LEDs available were small and have a limited watt input with less lumens per watt which determines the brightness, the CFL bulb was bought.

		
	LED	CFL
Average Life	45,000	10,000
Life Span	Very Long	Long
Wattage	2.5 - 16	3 - 120
Energy Usage	Low	Low
Price	High	Medium
Lumens Per Watt	45	60
Color Temp	2700k - 6500k	2700k - 6500k

Figure 29 - Comparison

Hence after careful comparison and consideration that the bulb has to be put in open areas where it will be utilized for parking spaces, CFL bulb is chosen for the application in the Smart parking system. Further, the specifications of the chosen bulb is given below in the table. (Anon, 2000)

<u>Data</u>	<u>Value</u>
Frequency	50 Hz
Rate Wattage	18.0 W
Voltage	240V
Average Life	10,000 Hours
Color Temperature	2700 K
Lumens	1200

Relay

There are different types of relays available in the market. Single relays of 5V, 12V, etc and then other are the relay boards which come with in-built transistor and resistor and an alternative way to make a circuit of relay. The figure below shows a normal relay and its pin diagram. NO port is for normally open which means the appliance will be on irrespective to the switch, whereas NC port is the normally closed port referring to the off point switch.

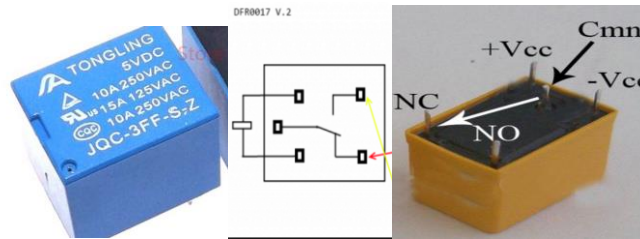


Figure 30 - Single relay

The second type of relay available is the relay board, which is basic relay mounted on a board and it is connected with a transistor and diode at the input end. The board is shown in the figure below and the circuit can be shown as well.

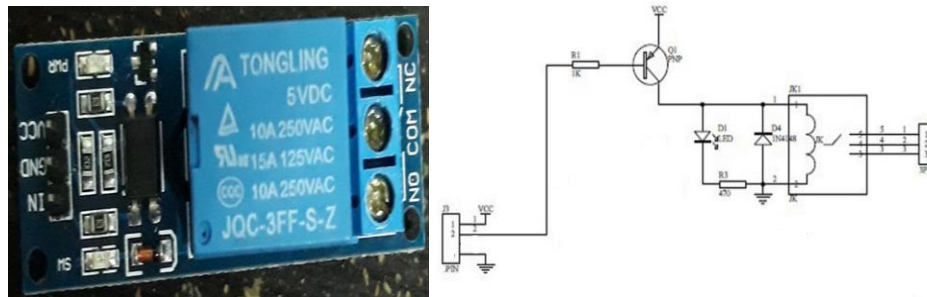


Figure 31 - Relay board with built transistors

The difference between the two types of relays are merely its circuit and the ability to handle external current drawn. The need to connect the relay to an external supply and not from the Arduino is the key reason using the relay board which has a transistor (DP817C) built in. In this relay board at the input side the ground and the Vcc can be supplied from elsewhere. The Relay board was used in the Smart parking system and the model was 5V JQC-3FF-S-Z.

Another alternative was to build up a relay, which can be shown in the circuit below

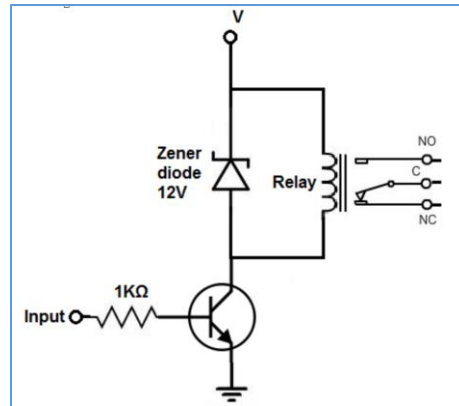


Figure 32

In this case a transistor is added for driving the relay which will take in less amount of current, by passing little required current to the base. This input pin is the digital pin from the micro controller and gives pulses, and hence when there is no pulse there is nothing in the coil, and vice versa. There is not much difference in the built already board relay and a separate process of making it, both limit the excess current coming in from the input and take in minimum amps to get started.

The specifications of the relay is shown below:

<u>Data</u>	<u>Value</u>
Coil Power	360mW
Nominal Voltage	5Vdc
Maximum Voltage	6.5Vdc
Coil Resistance	60Ω
Maximum Switching Current	10A
Operating Temperature	-40°C – 85°C
Weight	10.0g

Testing and Connection of CFL Bulb

For the regular LED's when connected the one of the smaller pin was easily connected to the ground of the breadboard and the other pin was connected to the trigger pin directly.

Since the higher watt (power) of CFL bulb was chosen it was clearly essential that the connection could no longer be taken from the Arduino or small battery supply. The two bulbs, one for entry and one for exit had to have an external source supply and hence they were put to on a bulb holder which was given a wired plug to be inserted to the socket supply. Now the main key point here is the connection of the bulb to the trigger pin of the controller which shall control the bulbs status. For this reason two relays of one channel each were taken for both the bulbs. The task of this relay which takes in the Vcc, ground from the battery which is giving 5V using a voltage regulator, the trigger pin connection from the microcontroller and on the other hand take a single wire of the plug and a wire from the bulb holder. This relay works as a switch, by connecting the bulb to the plug only when the trigger is high. Pins of relay were NC and common. Below is the connection shown for one single bulb and its testing.

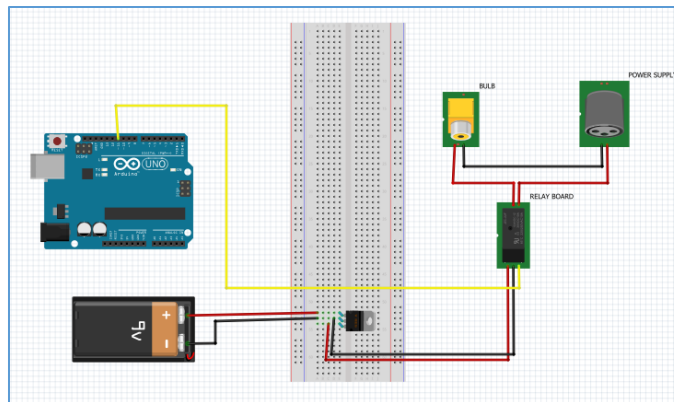


Figure 33 - Testing connection of CFL bulb

The code for testing is given below, the code shows the definition of the digital pin at the microprocessor, then the pulses sent are high and low for 15000 ms, this will give ideal time for testing at both the ends.

```

#include <Wire.h> // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>
#include <LiquidCrystal.h>
#include <Servo.h>

const int ledRout = 11;

void setup()
{
    pinMode(ledRout, OUTPUT);
}

void loop()
{
    digitalWrite(ledRout, LOW);
    delay (15000);
    digitalWrite(ledRout, HIGH);
    delay(15000);
}

```

Figure 34 - code for testing

For better understanding of CFL bulbs a plot of the current prices and the earlier prices can be done to know better advantages.

Testing the bulb was done by connecting it to the power supply, this power supply was AC supply directly from the socket which can be checked using the multi-meter. The bulb was placed on the bulb holder and one line was open circuited to find the current flowing through the bulb. This gave an idea of the resistance. Similarly other bulbs with varying Watts were measured and the resistance was found out. The plot of watts vs resistance and watts vs current gives a fair idea about how CFL bulbs operate and work with how much power. The relay's digital pin was first open circuited to find the current it is drawing from the pin so as to know whether it is exceeding the given limit of 10mA or not. Further for general information about the relay the Vcc from the voltage regulators output to relay board was open circuited to find out the current when the bulb was on and also when the bulb was off.

Code of CFL – For the system

The code for using the Bulb looks as such:

Defining the pins for the two Bulbs, the pins with the wires have to be connected to the relay from the microcontroller.

```
const int ledRout = 11;
```

```
const int ledGout = 12;
```

.....In the setup.....

```
void setup()
```

```
{
```

```
.....
```

```
.....
```

```
}
```

```
.....
```

```
pinMode(ledRout, OUTPUT);
```

```
pinMode(ledGout, OUTPUT);
```

```
pinMode(ledRin, OUTPUT);
```

```
pinMode(ledGin, OUTPUT);
```

```
.....
```

```
}
```

For the logical on and off of the bulb. The digitalWrite gives the signal from the pin to the destination to whether HIGH (on) or LOW (off) the bulb.

```
if (.....)
```

```
{
```

```
.....
```

```
digitalWrite(ledGin, HIGH);
```

```
digitalWrite(ledRin, LOW);
```

```
.....
```

```
}
```

```
else if (.....)
```



```

{
.....

digitalWrite(ledGin, LOW);

digitalWrite(ledRin, HIGH);

}

```

Since there are two bulbs used for the two points entry and exit, the same fragment can be pasted for the other bulb as well.

LCD

For a visible display of text, or count, or any indication we need a display. This display can be the computer, an external monitor or an LCD. A display is almost the most important part of an embedded system. Comprising of liquid crystal molecules which control the passage of light and take the pattern to display is what an LCD is. The Liquid crystal display has its own backlight which can be controlled using the software. The backlight uses Cold Cathode Fluorescent Lamps (CCFL). Type 1 LCD which comes with 16x1 display unit, this LCD has 1 line on which 16 characters can be displayed, type 2 is the 16x2 with two rows. The basic model of the LCD used with Arduino is HD44780, this HD44780 controls the pixels which is present at the back.

The LCD 16x2 comes in two different forms, one with the I2Cbus interface one without it. The main difference between the LCD with an I2C interface and the non interface is that the I2C interface one uses only 2 microcontroller pins to set the LCD working and the other 2 pins go to ground and VCC. Whereas, the non I2C interface one requires all the 16 or less at least 6 pins to be configured with the controller.

SDA	A4	I2C data line
SCL	A5	I2C clock line

Figure 35 – Pin (Circuitar.com, n.d.)

However the LCD without the I2C interface can be soldered with the Interface by buying external bus. Almost all the LCD's take 5V which can directly be supplied from the microcontroller or external battery or supply. The display of the LCD works by first completely

utilizing the row 1 and then going next line, also the cursor can be set using the programming. (Datsi.fi.upm.es, 2008). The I2C (Inter-IC) by definition is a bi-directional two-wire serial bus that links the ICs. The SCL pin is the clock pin which is low then only the data with SDL data pin take place.

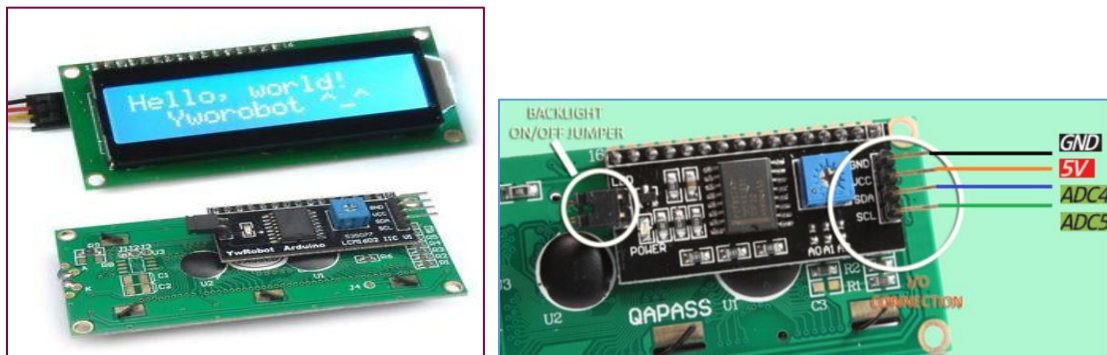


Figure 36 - LCD

In this regard referring to the smart parking system it is essential to note that to reduce complications and wiring mess, the LCD with I2C was used. This saved time and made the connections easy. The specifications of the LCD can be tabulated below:

<u>Data</u>	<u>Value</u>
Working Voltage	4.5 – 5.5 V
Supply Current	0.8 – 1.5mA
Working current	34mA
Characteristics	16 Characters x 2 Lines
Module Dimensions	80.0x36.0x14.0
Operating Temperature	-20°C - 70°C

Testing and Connection of LCD

The connection was made purely to display the first initial message when the system is started and then give the count of the slots available in the parking system. Using the I2C mounted LCD it has become relatively easy to connect the LCD to the microcontroller.

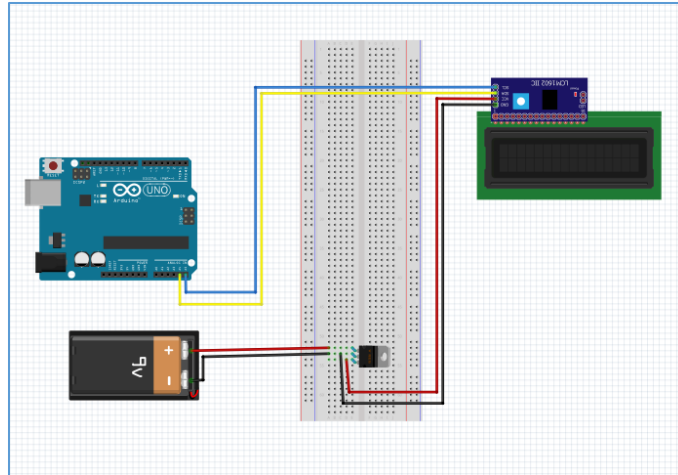


Figure 37 - Connection LCD

The only 4 connections shown are the VCC and Ground which are taken from an external battery. The other two pins of the I2C pins are the SDA and SCL pins, the data and the clock pin. The SDA pin goes to A4 and SCL pin goes to the A5 pin of the microcontroller. The code to test the LCD is given below:

```
#include <Wire.h> // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>
#include <LiquidCrystal.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup()
{
  lcd.begin(16,2);
  //setting cursor position to start of first line of the LCD
  lcd.setCursor(0,0);
  lcd.print("    LCD");
  lcd.setCursor(0,1);
  lcd.print("    TEST");
}

void loop()
{
}
```

Figure 38

The code shown in the above figure will print LCD TEST in the two lines of the LCD. The alternative way could be to put it in the void loop section and introduce a delay. The address specified in the initial lines has to be found out separately as each LCD has a different address.

For the testing the I2C mounted on the back of the LCD has a potentiometer which controls the cursor and its settings. This potentiometer controls the flow of supply current as well. When kept at ideal condition the potentiometer to view the cursor and the text the current drawn can be seen by open circuiting the voltage supply to the LCD. Further the resistance can be measured across the LCD while the voltage is kept constant and supply current can be found.



Figure 39 - I2C

The code when uploaded to the system can show the Text printed.

Code for LCD – For the system

The code for the LCD can be relatively generated after getting the I2C address as it's different for all the different I2C LCD's. That code and relative finding of the address is given in the program development section. The code for LCD is generated and can be used as such:

The header files Liquid crystal I2C has to be used to use the library for using I2C mounted LCD.

```
#include <LiquidCrystal_I2C.h>
```

```
#include <LiquidCrystal.h>
```

```
//Setting up the LCD with I2C module
```

```
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C address
```

Further we have the initial message display, the word which comes up in the display SMPS means Smart Parking System. The set cursor command is used to set the cursor from 0, 0 means first column and 1st row. The second one 0, 1 means the 1st column and the next line the 2nd row. lcd.print command is used to give instruction to the LCD to show the following words.

```
{
```

```

Serial.begin(9600); // Used to type in characters

lcd.begin(16, 2); // initialize the lcd for 16 chars 2 lines, turn on backlight

{

  lcd.setCursor (0, 0);

  lcd.print("Welcome To");

  ....

  lcd.setCursor (0, 1); // go to start of 2nd line

  lcd.print("SMPS");

  ...

}

```

This command after displaying initial message leaves the LCD on and its backlight lit.

```

lcd.backlight();

```

The clear command is used to clear off anything which has remained on the display. Further as explained earlier the command to set cursor and print the information after computations and show how many parking slots are available.

```

lcd.clear();

lcd.setCursor (0, 0); // go to start of 1st line

lcd.print("No. of Lots: ");

lcd.setCursor (0, 1); // go to start of 2nd line

lcd.print(slots);

}

```

Complete Combined Outputs

All the outputs parts and their connections which were defined and described with individual circuits are now to be brought together after testing. All the connections when made for one bulb and one servo look like this:

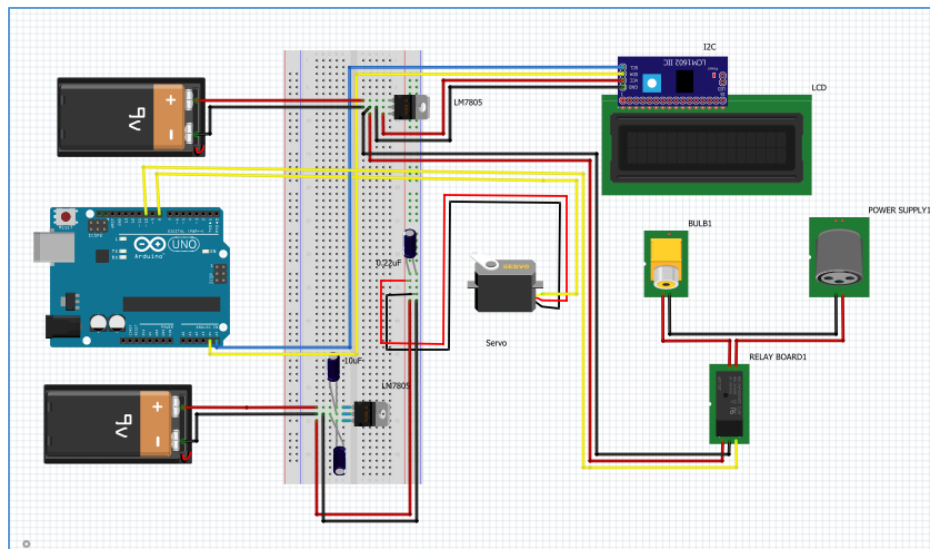


Figure 40 - Combined outputs connection

After connecting all the outputs which is two bulbs, two servos, and for proper usage according to the smart parking system it is essential to combine them with the input side. And then check with results of the output while they were working with the inputs.

Results Discussion on Output Interface Design

According to the connections which were shown in the previous section the outputs were put together and run with the inputs and also the outputs were checked separately.

The checking of the servo motors for the current drawn at different positions was done. Since the motor moves continuously it had to be given a huge delay so see the current drawn at specific angles. The table and the graph can be plotted for current drawn at different positions below.

<u>Position (degrees)</u>	<u>Current Drawn (mA)</u>
0	30.78
22.5	24.23
45	15.44
67.5	10.67
90	3.56

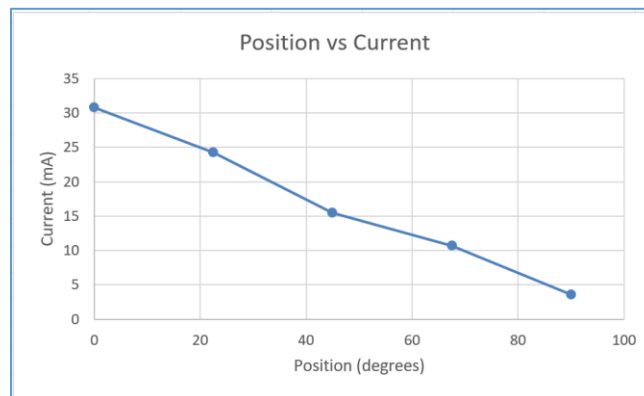


Figure 41 - Servo Position vs Current

It can be seen effectively that by each degree of closeness to the destination the current drawn from the servo motor is reduced. This means that the starting point of the servo motor it draws around 30mA from the circuit which is above the criteria of 10mA and hence external battery has to be used for using servo motors.

Further the voltage when supplied to the servo motor was up to its requirement it works fine. The varying voltage when done made the motor to draw more current as it needed to cope up with the requirements. The voltage and current reading are seen to be:

<u>Voltage (V)</u>	<u>Current Drawn (mA)</u>
5	3.56
5.2	3.72
5.5	3.91
5.8	4.12
6	4.27

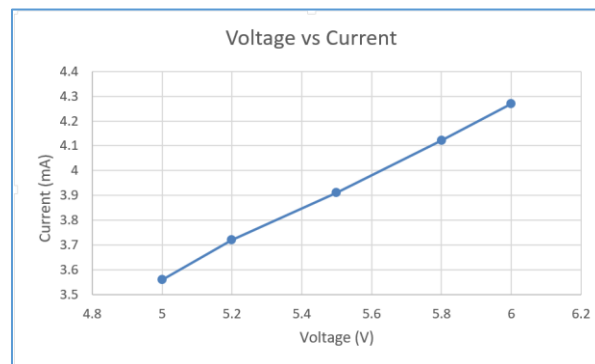


Figure 42 - Servo V vs I

During the testing of servo motors several difficulties were faced, the first and foremost connecting the servo motor directly to the Arduino and checking. The current drawn was too high, and the set limit maximum it can draw was 10mA, and hence for this reason external battery was used. Another issue the battery's voltage was too high for the servo motor and had burnt one of it when it was directly connected thus it had to be controlled; it can be done using a resistance or a voltage regulator, the latter being a good choice due to its exact dissipation of voltage. Some batteries when used of 6 or 7 volts could generate enough current even though the battery were controlled to give the servo motor. Hence the higher volts battery had to be used.

The implementation of the CFL bulbs for testing generated results for the bulb which were based on basic readings at first. While the bulb was in the bulb holder detached from the circuit and only directly connected to the power supply socket, the voltage across the bulb was read 248V and the current after open circuit was read 82.1mA.

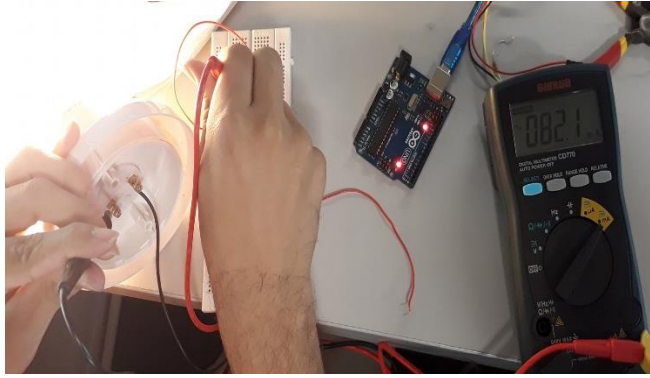


Figure 43

$$V = 248 \text{ Volts, } I = 82.1 \text{ mA,}$$

Thus,

$$P = V \cdot I = 20.3 \text{ W.}$$

This gives us the idea for the resistance of the bulb which can be given as;

$$R = V / I = 3.02 \text{ k}\Omega.$$

The tabulated values for different watts of bulbs while taking in 240V to find the resistance and consumption of current is given below:

<u>Power (W)</u>	<u>Current(mA)</u>	<u>Resistance(kΩ)</u>
6.55	27.2	8.82
10.03	41.79	5.7
16.79	69.9	3.45
19.26	77.7	3.20
26.67	108.33	2.215

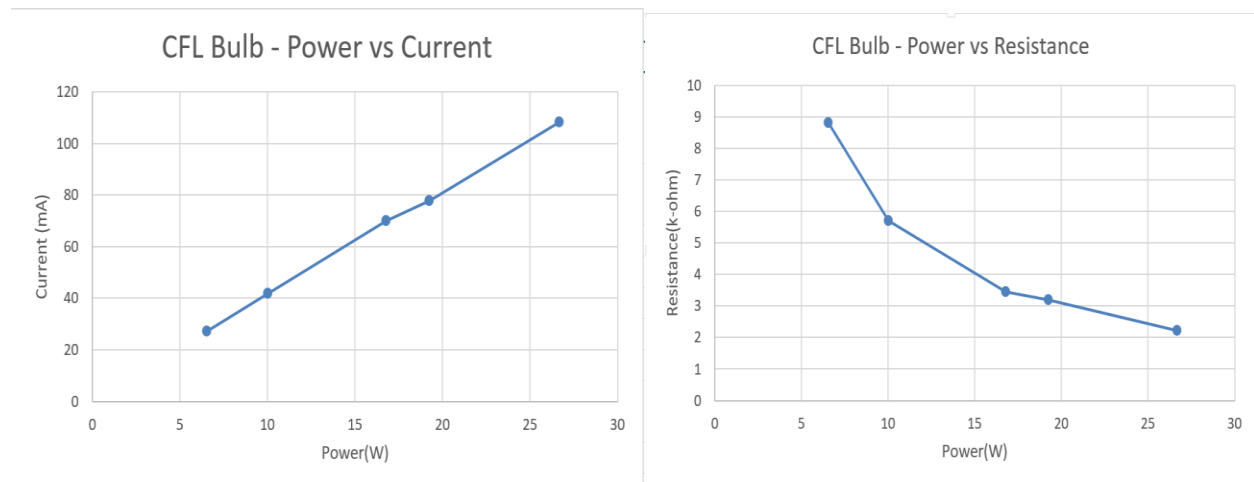


Figure 44 - a) W vs mA b) W vs k-ohm

From the above result we can infer that, higher the power higher will be the current drawn and less will be the resistance. The main purpose of higher wattage bulbs is to give more brightness and hence the current is required to flow for achieving it. The higher the wattage the more was the supply needed, and it became increasingly difficult to give such a high voltage using battery and thus AC supply to the bulb had to be used, as that can give enough to light up the bulb. At first using of incandescent bulb was a choice but they burn and consume lot of power thus we moved to CFL for real time.

The current drawn by the relay was 3.7mA when the pulse was low given to the board by the processor as shown in the figure below.

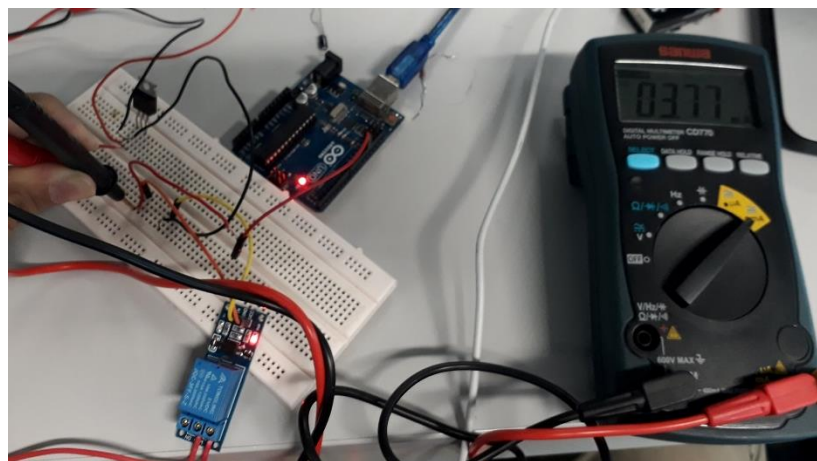


Figure 45

When the pulse was high it read 77mA, which is similar to when the switch is not connected as shown earlier. The relay single was used but then it was drawing lot of current, thus the board relay with a transistor had to be used. Even if this relay was connected to the micro controlled it was drawing only 4mA but the issue rising was, when the pulse was low to the relay the bulb was still pulling around 80mA in its off state from the other supply, since it is essential that the relay while switching doesn't harm the processor battery to turn on the relay was used again with a voltage regulator.

As for the LCD testing the given voltage was measure as 4.93V from the regulator taking in supply from 9V battery. It was then open circuited to find the current as 25.89mA while viewing the text on the LCD. The resistance at that particular instant can be calculated as $0.19k\Omega$.

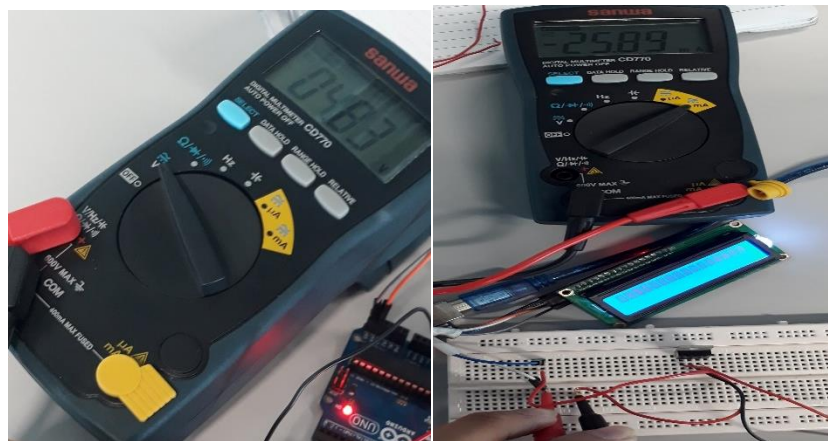


Figure 46

The resistance of the LCD can be varied as explained earlier. Different resistances using the potentiometer were set and the supply current was calculated by keeping the voltage supply constant at 5V.

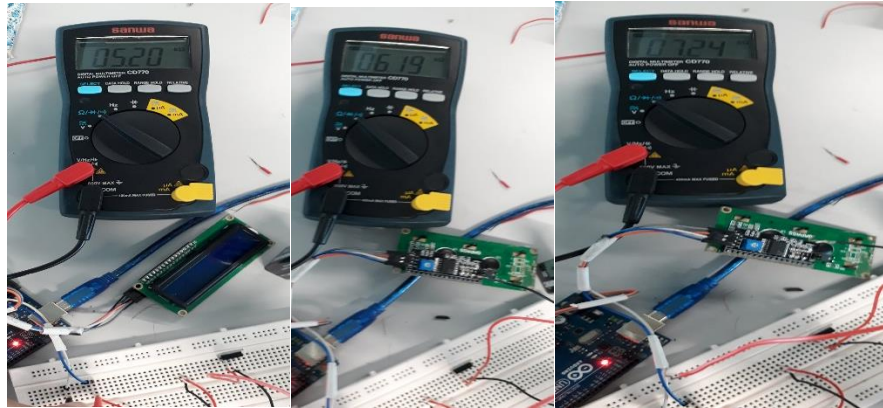


Figure 47 – Testing Results

<u>Resistance Measured (kΩ)</u>	<u>Supply Current Calculated (mA)</u>
5.20	0.96
5.77	0.87
6.19	0.80
6.73	0.74
7.21	0.69

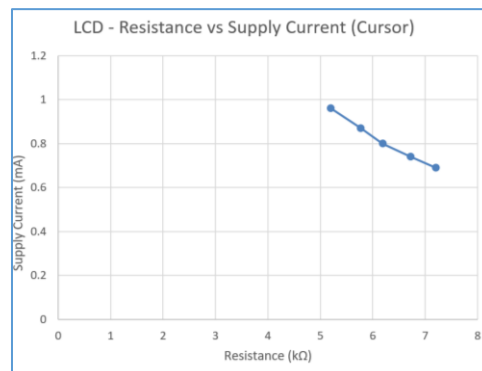


Figure 48

From the above graph we can see how the cursor potentiometer behind the LCD in the I2C module dims the cursor by limiting the current flow. However, the testing of LCD had few challenges which started with using I2C and the current drawn by the LCD. The code developed

for testing gave errors and hence the library was missing which has to be put for using I2C. Further the LCD drew more than 10mA of current and thus it was essential to connect it to the external supply. Although the current could be controlled but it then reduced the display of the LCD, therefore it was best to use battery and then control the voltage accordingly for the LCD.

The overall combinations of the all the outputs when joined with the inputs for the final prototype of the parking system they did not disappoint and worked effectively. The results of the output while using with the system can be shown.

The figure below shows how the CFL bulb works in the parking system by being a proper real time applicable object using the relay board. The connections were made according to how it is shown in the testing part. The bulb lights up when the car is sensed and the barrier is open for the car to go in. The pulse is sent to the relay which is acting as a switch and thus the bulb turns off once the barrier is down and car is gone.

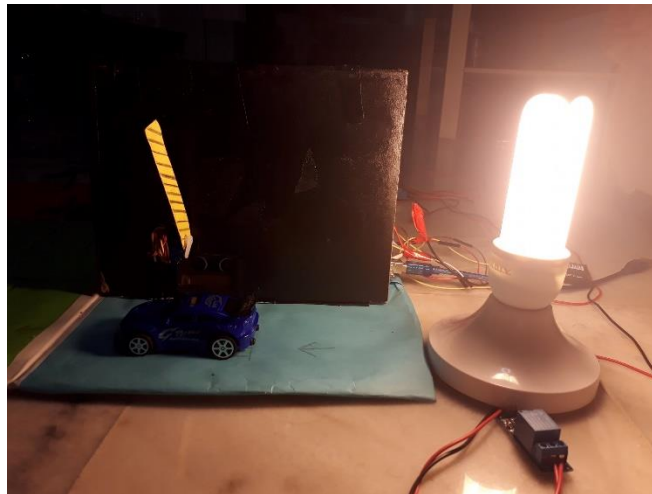


Figure 49 –Working Bulb

The next figure shows how the servo motor is working in the system. It is sent a signal by the processor to go up and hold till the car is passed, once it passes the servo motor receives the pulse to go down to normal angle. The angle is fed into the servo motor to move to a particular height and return in very less time. Limiting the number of slots and the logic is fed into the servo motor and thus it doesn't lift up the barrier.

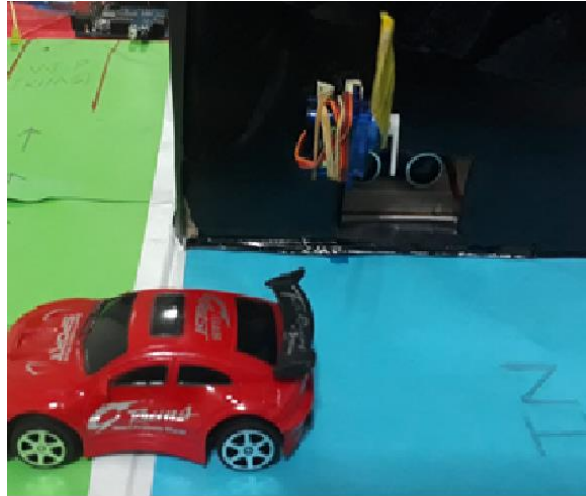


Figure 50 - Working Servo

The display LCD used in the system using the I2C channel simplified the process and was effective. As it can be seen in the figure below how the no of lots appear on the LCD screen. Initially when the system is turned on the LCD screen shows WELCOME TO SMPS, the SMPS refers to smart parking system. The SDA and SCL pins from the microprocessor transfer the data to the screen for showing this. Next once the sensors start sensing the slots available and decrement with every entry. This in-turn shows the counter at the LCD for the drivers to see and know. The count is given in the code and that is being printed on the LCD.

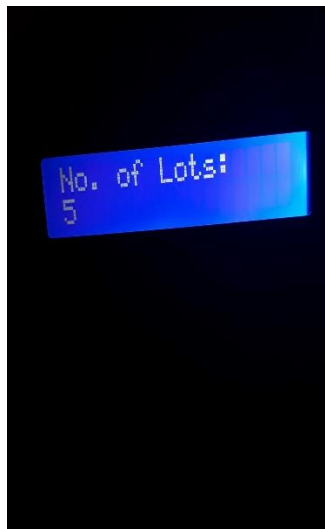


Figure 51 - Working LCD

Since the completion of the system looks fair extremely relevant to the current situation as discussed in the problem statement. There were difficulties which came in to the picture while developing the whole system. The problems were from scratch and effected the output of the system mainly. For each component of the output the issues and the possibilities were sort while their own testing time. However there were few which turned up after combining all the outputs with inputs. This resulted in huge messy connections had to be simplified by taping and so on. There were three to four different batteries which were used to power up while there were wires going to the power supply as well from the bulb due to real time relay connection. This can be reduced by using a single high volts battery but different higher voltage regulator to adequate the power for each component.

Program Development

Microcontroller History

During 1970 & 1971 when Intel was working on inventing the world's first microprocessor, Gary Boone & Michael Cochran were two engineers of Texas Instruments were working on quite a similar concept & as a result microcontroller was invented. They built a 4-bit microcontroller TMS 1000 that had ROM & RAM incorporated in it. From 1972 to 1974, TI used TMS 1000 in its calculator products. TMS 1000 having different RAM & ROM configurations were made available for the electronics industries to buy in 1974. Almost 100 million TMS 1000 microcontrollers were sold within 10 years period.



Figure 52: TMS 1000

Early microcontrollers were essentially microprocessors with memory for example, RAM & ROM incorporated within. There was a time when people used an Abacus just to perform simple calculations. Bulky wooden frames were utilized just to do some basic arithmetic operations. Vacuum tubes with magnetic drums made an entry as the 1st generation computers but they were replaced by transistors followed by integrated circuits as an aid for the 3rd generation computers. The revolution came when microprocessors were created where thousands of Integrated circuits were built in. A microprocessor lies at the heart of most computers & does all the work within.

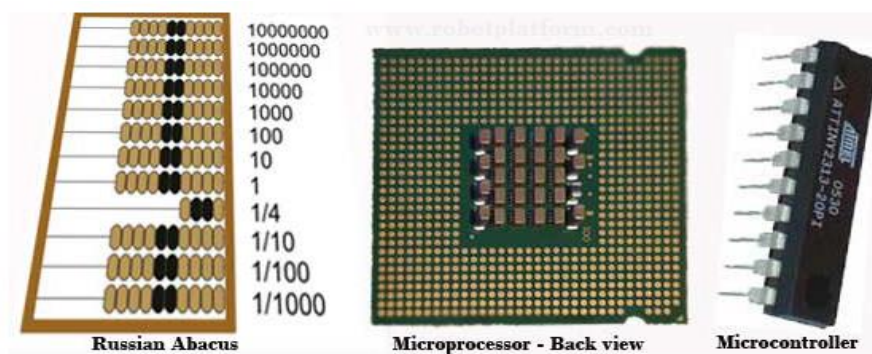


Figure 53: Microcontroller

With the passage of time, a lot of advancements were made in the microcontrollers & they were specified to perform a specific task or to control a specific application for example engine system in vehicles, mobile phones etc.

Intel Microcontrollers

The most important microcontrollers produced by Intel are the 8048 & the 8051 microcontrollers. In 1976, Intel developed their very first microcontroller 8048 and it was incorporated in the IBM pc keyboard as a processor. Intel 8048 is actually a member of Intel MCS-48 family of 8-bit microcontrollers. The MCU has on-chip clock oscillator, 2-8 bit timers, 27 I/O ports, 64 bytes of RAM & 1 KB of masked ROM. Then IN 1981, Intel developed 8051 microcontroller. It was referred as system on a chip as it had 128 bytes of RAM, 4K byte of on-chip ROM, couple of timers, a serial port & 4 ports all on single chip. Later when it became widely popular, Intel allowed other manufacturers to market & make different flavors of 8051 microcontroller. In simple words, it means that if a program is written for one flavour of 8051, it can run on other flavors as well regardless of the manufacturer.



Figure 54: Intel 8051 microcontroller

Microcontrollers have been used for ages now & their impact is deep. Normally they are embedded within devices so that action of these devices can be controlled. Typically a microcontroller is utilized for 3 basic reasons:

- 1- Receive input using sensors, human intervention etc.,
- 2- Store the input and process it into a set of actions.
- 3- Apply the processed data for some other actions that goes as an output.

Arduino

Arduino is the world's leading open-source hardware & software ecosystem. This company offers a range of software tools, hardware platforms & documentation enabling almost anybody to be creative with technology (Arduino.cc, 2017). Arduino is a single board microcontroller that is designed to make applications, interactive controls or environments easily adaptive. As far as hardware is concerned, it consists of a board designed around an 8-bit microcontroller. Arduino hardware part is basically based on a simple programmable microcontroller board along with the software part being the IDE (Integrated Development Environment) that is used to write & upload codes to the physical board. Arduino doesn't need a separate piece of hardware to load the code onto the board, a USB cable can simply be utilized unlike other previous programmable circuit boards. Arduino can create interactive objects by having the capacity to take input from a variety of sensors & switches & controlling an assortment of motors, lights & other outputs. Arduino can be connected to the IDE and can be controlled accordingly. Actually the connection of the wires on the microcontroller board is based on the Arduino programming. Current models feature things like a USB interface, analog inputs & GPIO pins that allow the user to attach additional boards. Some of the largest semiconductor companies have jumped into the Arduino space such as Cypress, STM, Texas Instruments, Freescale, and incumbent Atmel.

There are some major benefits of using Arduino that are as follows:

- 1- Cheap- Arduino boards are inexpensive when compared to other microcontroller platforms. The least costly adaption of the Arduino module can be assembled by hand, & even the pre-assembled Arduino modules cost under \$50.
- 2- Simple- IDE software is easy to use for beginners yet flexible for advanced users.
- 3- Cross-platform- Arduino IDE software runs on Windows, Mac & Linux operating operations. As far as other microcontrollers are concerned, most of them are only linked to Windows.
- 4- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers.
- 5- Open source and extensible hardware – Arduino is basically based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. In order to help experienced designers

so that they can make their own version of the module, plans for the modules are published under a Creative Commons license. On the other hand inexperienced users can build the breadboard version of the module to understand the working & in order to save money (Learn.sparkfun.com, 2017)

ARDUINO C GRAMMAR/LANGUAGE

The aim of Arduino is to provide flexible, easy to utilize hardware & software but the difficulty lies in the Arduino programming language that controls the hardware & is controlled by a set of C/C++ functions. Arduino grammar is based on C/C++. C programming language features were derived from an earlier language called BCPL (Basic Combined Programming Language). It was invented for implementing UNIX operating system. In 1978, Dennis Ritchie & Brian Kernighan invented C programming which was later changed to C++ in early 90s by Bjarne Stroustrup by introducing concepts that were object oriented. C++ is much more advanced than C programming. Even C++ and C are sharing lots of its concepts & syntax; still both languages are different from each other. In order to write a C++ program, one or more source files are required as it is a compiled language. The source files have to be compiled in order to get a program that can be executed via a program called compiler. This process is known as compilation. Compiler like Keil uVision can be used.

IDE

IDE stands for Integrated Development Environment. It is basically a software application that provides comprehensive facilities to computer programmers for software development. Typically an IDE consists of a source code editor, build automation tools & a debugger. Arduino utilizes IDE in order to open, create & edit programs. The Arduino board with imbedded microcontroller will behave according to the program written.

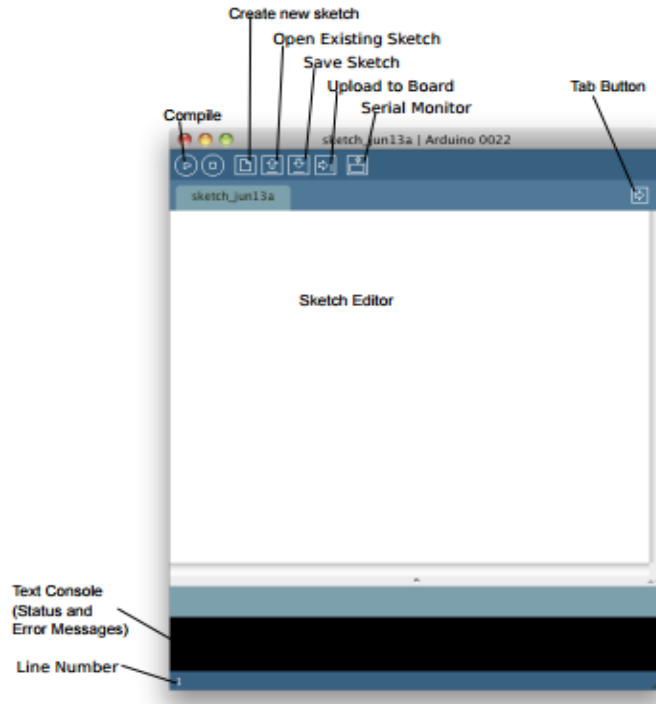


Figure 55: IDE interface

IDE features:

- Verify – Checks the code for errors compiling it.



- Upload- It basically compiles the code & uploads it to the configured board by using a USB cable.



- Create new sketch/Program – A new window is opened in order to write a new program.



- Open Existing program – Basically it presents a menu of all the sketches in your sketchbook.



- Save – Saves the program



- Stop – Stops the process of compilation.
- Compile- Converts the program into instructions so the board can understand.

Serial Monitor

- Tab Button – This button allows the user to develop number of files within a program.
- Editor – Where the program is written and can be edited as well.
- Text console – This actually displays what IDE is doing & error messages are also displayed.
- Line Number – This basically helps in detecting the line number where error has occurred.

ARDUINO STRUCTURE

Arduino boards are used for making different type of engineering projects & applications. Arduino board was designed in the Ivrea Interaction Design Institute for students without a background in programming & electronics. All boards are entirely open-source, allowing users to build them separately and finally adapt them to their exact needs. Arduino boards are highly recommended as they are used to build projects, from daily objects to compound scientific instruments. Arduino boards come in many varieties that can be utilized for different applications. Certain boards are a bit different from the figure illustrated below but most of the Arduino boards have these components in common.

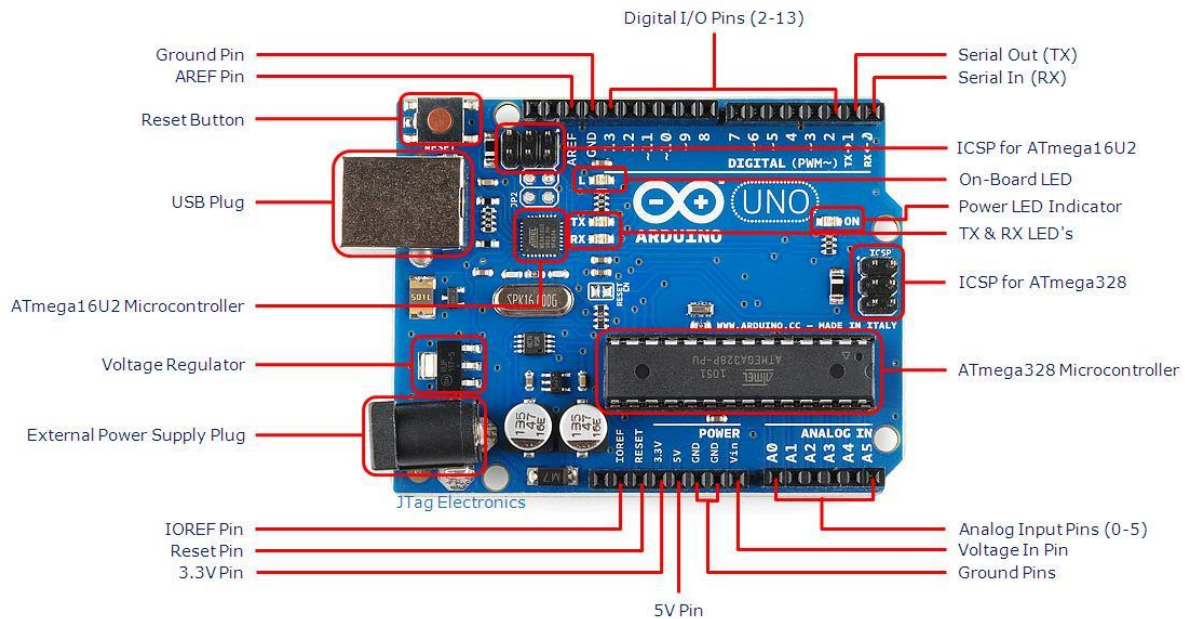


Figure 56: Arduino-Uno-R3

As far as the proposed design is concerned Arduino UNO is selected to work with. It is most basic & commonly used microcontroller. It works in such a way that for a given system the relation between input & output is required to be defined. One of the major reasons for selecting is because of its sophistication in managing devices & different components effectively. Arduino microcontroller is very flexible in terms of analog capability such as reading sensors. It is much easier to deduce & respond to a variety of sensors with Arduino as it offers beneficial repeated series of command & respond to sensor.

In order to do coding, the very first thing that is supposed to be done is to study about the microcontroller that is used for the proposed system. Arduino board is the best board to get started with electronics & coding. The different components on the Arduino board will be discussed below just to have a rough idea.

Power USB:

Arduino board is powered by utilizing the USB cable from the computer. The only required thing is to connect the USB cable with the USB connection.

Power

Arduino boards can also be powered directly from AC main power supply just by connecting it to the Power.

Voltage Regulator

The basic function of Voltage regulator is to control the voltage that has been provided to the Arduino microcontroller & stabilize the DC voltages utilized by the processor.

Crystal Oscillator

Arduino calculates time by utilizing the crystal oscillator. The number that has been printed on top of Arduino crystal 16.000H9H which tells the user that the frequency is in Hertz or MHz.

Arduino Reset

User can reset the Arduino board just by clicking the red button.

Pins (3.3, 5, GND, Vin)

- 3.3 V – It actually supply 3.3 volt (output)
- 5V – It actually supply 5 volt (output)
- Ground- This pin is actually used to ground the circuit.
- Vin – This pin can also be utilized in order to power the Arduino board by using an external power source.

Analog pins

Arduino board consist of 5 analog pins (A0-A5). These pins actually read the signal from analog sensor for example temp sensor & then convert it to a digital value so that it can be read by the microprocessor.

Main microcontroller

Microcontroller is basically a brain of the board. Each and every Arduino board comes with its own microcontroller. The main integrated circuit is little bit different from board to board. User must know that what IC his/her board has & the information is given on the top of the IC.

Power LED indicator

When Arduino is powered this LED lights up in order to indicate that board is powered on correctly.

TX and RX LEDs

Two labels can be found on Arduino board called as TX (transmit) & RX (Receive). They can be found in 2 places on Arduino board. First, they can be found at the digital pins 0 & 1 in order to indicate the pins that are responsible for serial communication whereas the Second TX & RX, TX led lights up with different speed when sending the serial data. RX flashes during the receiving process.

Digital I/O

Arduini Uno has 14 digital input/output pins and out of which 6 can be utilized as PWM output. These pins can work as input digital pins in order to read logical values or as digital output pins in order to drive different modules for example relays, LED's etc.

DEVELOPMENT

From information requirements to the final implementation, the system development cycle is an ongoing process. When the input and output components are connected to the arduino microcontroller in order to collect data from the environment & control different outputs, a set of programming logic is developed to interface inputs & outputs as well. Program development should include all the factors that influence the input & output devices that are connected to the system. As far as the proposed system is concerned, two separate microcontrollers are used for both inputs. Inputs with their outputs are tabulated below:

INPUT DEVICES	OUTPUT DEVICES
<ul style="list-style-type: none">Couple of HC-SR04 Ping Sensor	<ul style="list-style-type: none">LEDsServo Motors

INPUT DEVICES	OUTPUT DEVICES
---------------	----------------

<ul style="list-style-type: none"> • Photoelectric IR Sensor 	<ul style="list-style-type: none"> • LED
---	---

Arduino is basically open source computer hardware & a software company. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on C++ but in a simplified form.

Logic

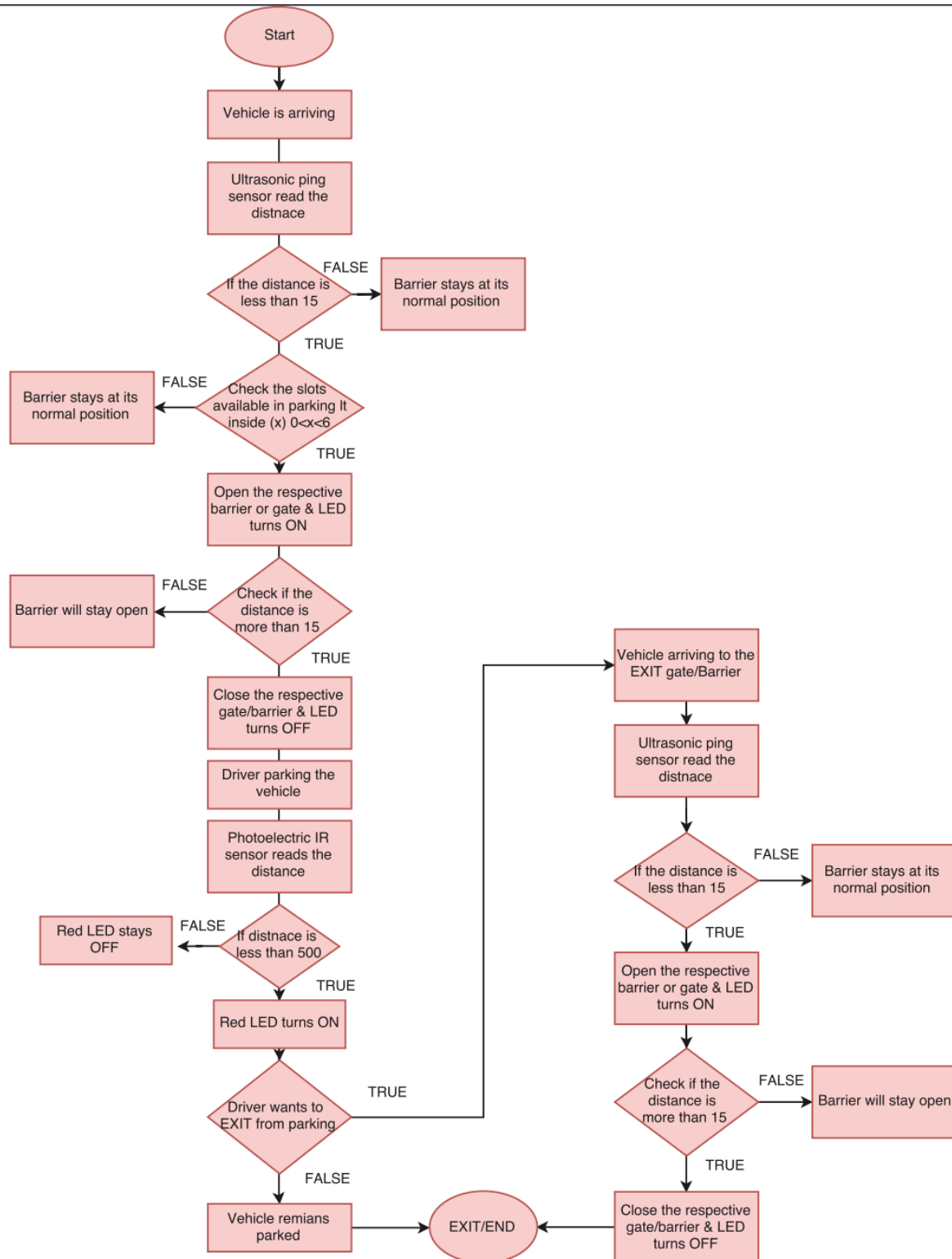
Programming logic is basically a fundamental construct that is applied to the computer science in a variety of comprehensive ways. Programming logic involves logical operations on hard data that works according to logical principles & quantifiable results. As far as the programming logic for the proposed system is concerned, it can be divided into three parts. The very first part includes data collection from the input devices. Couple of ultrasonic ping sensors are utilized in order to obtain data from both sides (entrance and exit). In simple words ultrasonic ping sensors are basically collecting distance information data from vehicles. After receiving the input, the system is supposed to be programmed in such a manner that it can perform task & can make decision by using input data. The ultrasonic sensor is utilized in order to sense the presence of the vehicle and then only barrier will be pulled up so the vehicle can pass. In order to make the system more precise & accurate the system is programmed in such a manner that if a vehicle distance is less than 15cm then it is classified as close vehicle & then only barrier will be pulled up. The microcontroller is programmed in such a manner then when the vehicle is close, arduino microcontroller will send output signals to turn on the light (bulb) for the entrance side & to pull up the barrier using servo motors by assigning a certain angle value. When any vehicle comes close enough to be in the 'close' bracket, the no of slots is incremented when the vehicle is entering & the no of slots is decremented when the vehicle is leaving. The number of slots actually represents the number of slots that are available in the parking area. LCD monitor is utilized for displaying the number of slots.

As far as second input is concerned, one photoelectric IR sensor is used in order to obtain data. IR sensor is basically collecting distance information data from vehicles. IR sensor is used in order to sense the presence of the vehicle in the parking lot. IR sensor acts as a parking aid in

order to help the driver while parking the vehicle. Once the driver crosses the safety limit and enters the danger zone, a red colour LED lights up as an indication so that the user can know that if he continues to do he can bump his/her vehicle & can end up having a dent on his/her vehicle. The objective is to make a controller which is an aid to weak drivers while parking the vehicle.

Flow chart

The flow chart illustrated below shows an overview & the actual procedure that the system goes through in order to perform the specific task in the form of sequence of events that take place in the background of the main system. In addition it presents the overall structure by using flow chart. Since flow chart is the most convenient way of presenting the program design. The flow chart illustrated below shows the working of both sensors that may include Ultrasonic IR sensors & photoelectric IR sensors. This flow chart is designed to give a better view of the system that how both inputs work for the parking system.



Initially when car is arriving, ultrasonic ping sensor reads the distance. If the distance of the vehicle is greater than 15cm, availability of lots is checked. 6 slots are designed in the system. If any of the slots is empty, barrier will be pulled up making Led ON. If the slots are not available or parking is full barrier will not open. Ultrasonic keeps on sensing and as soon as the car has passed, ultrasonic ping sensor will sense the distance and as the vehicle has moved so the distance is more than 15cm and as a result barrier will close or come back to its original position making LED light OFF. In case the car has not moved, barrier will stay open until the car has passed. Now the second input of the system that is photoelectric IR sensor comes into action while the driver is parking his/her vehicle. Photoelectric IR sensor that is attached with the wall reads the distance. When the driver is trying to park his/her vehicle & the distance is less than 500 which means that the vehicle is in danger zone and as a result RED Led will switch that is basically an indication for the driver & when the distance is more than 500 Led light will remain off. Coming back to the first input when the driver wants to leave the parking lot or wants to exit. The very same procedure is repeated for exit as well as illustrated in the flow chart given above:

Pseudocode

The pseudo code for the program development is done in order to achieve the main objectives of the proposed system is shown below:

ULTRASONIC PING SENSOR

ALGORITHM 1: Connecting Pins and Defining Variables

```
#include all required libraries.

#define pin numbers for Ultrasonic.
#declare triggerpin[incoming, outgoing] = [2, 4];
#declare echopin[incoming, outgoing] = [3, 5];

#define LED pins.
#declare LEDs[incoming.greem,: outgoing.red] = [11 : 10];

#define the servos installed
#define servo.incoming
#define servo.outgoing

#set up the LCD with I2C monitor

#declare variable for slots
    #assign slots = 5;

#declare variables for ping sensor
```

ALGORITHM 2: Setup Loop

```

#define baud_rate for serial interface connection

#initialize the LCD for 16x2
    #assign LCD.backlight value = 1;

#for i = 0 and i < 3 (increment i)
    #turn LCD backlight on
    #wait
    #turn LCD backlight off
    #wait
    #print.lcd developers info
#end for statement

#attach servo.incoming to pin 6
    #assign servo.angle = 'closed'
#attach servo.outgoing to pin 7
    #assign servo.angle = 'closed'

#declare LEDs[incoming. green: outgoing.red] = [11 : 10];

#set pinmode for LED = OUTPUT
#set pinmode trig = OUTPUT
#set pinmode echo = INPUT

```

ALGORITHM 3: Main Loop

```

#clear trigger pin for incoming ping sensor
#set trigger.incoming = HIGH
#wait
#set trigger.incoming = LOW
#set echo.incoming = HIGH
#assign duration = echo.incoming
#record distance.incoming
#wait
#clear trigger pin for outgoing ping sensor
#set trigger. outgoing = HIGH
#wait
#set trigger. outgoing = LOW
#set echo. outgoing = HIGH
#assign duration = echo.outgoing
#record distance.outgoing
#wait

#print serial (distance.incoming)
#print serial (distance.outgoing)

#if (distance.incoming < 15cm)
    #increment slots & slots >0
    #set servo.incoming angle = 'open'

#else if (distance.incoming > 15cm)
    #wait
    #set servo.incoming angle = 'closed'
#end if statement

#if (distance.outgoing < 15cm)
    #increment slots & slots <6
    #set servo.outgoing angle = 'open'

#else if (distance.outgoing > 15cm)
    #wait
    #set servo.outgoing angle = 'closed'
#end if statement

#clear LCD display
#print.lcd 'slots'
#end

```

PHOTOELECTRIC IR SENSOR

ALGORITHM1: Connecting Pins and Defining Variables

```
#include all required libraries.
```

ALGORITHM2: Setup Loop

```
#Initialize Analog pin A0 as input to receive signal  
#Initialize Analog pin A1 as output for ground  
#Initialize Analog pin A2 as output for Vcc  
#Initialize the pin number for LED  
# Supply 5V to the terminal of sensor  
#Connect sensor ground with Arduino ground  
  
#define baud rate for serial interface connection
```

ALGORITHM3: Main Loop

```
#Print voltage of output pin resolution (1-1024)  
    #wait  
#if (distance.incoming < 500)  
    #LED light turn ON  
  
#else if (distance.incoming > 500)  
    #LED light turns OFF
```


CODE & ALGORITHM EXPLANATION

ULTRASONIC PING SENSORS

Since the project is made of two parts & both the parts are not interrelated so there shall be two different codes for both the inputs. Firstly ultrasonic Ping sensor code will be explained followed by the second input of the system which is Photoelectric IR sensor. A separate block diagram is designed in order to give a better understanding about the system. The entire code will be based on this block diagram:

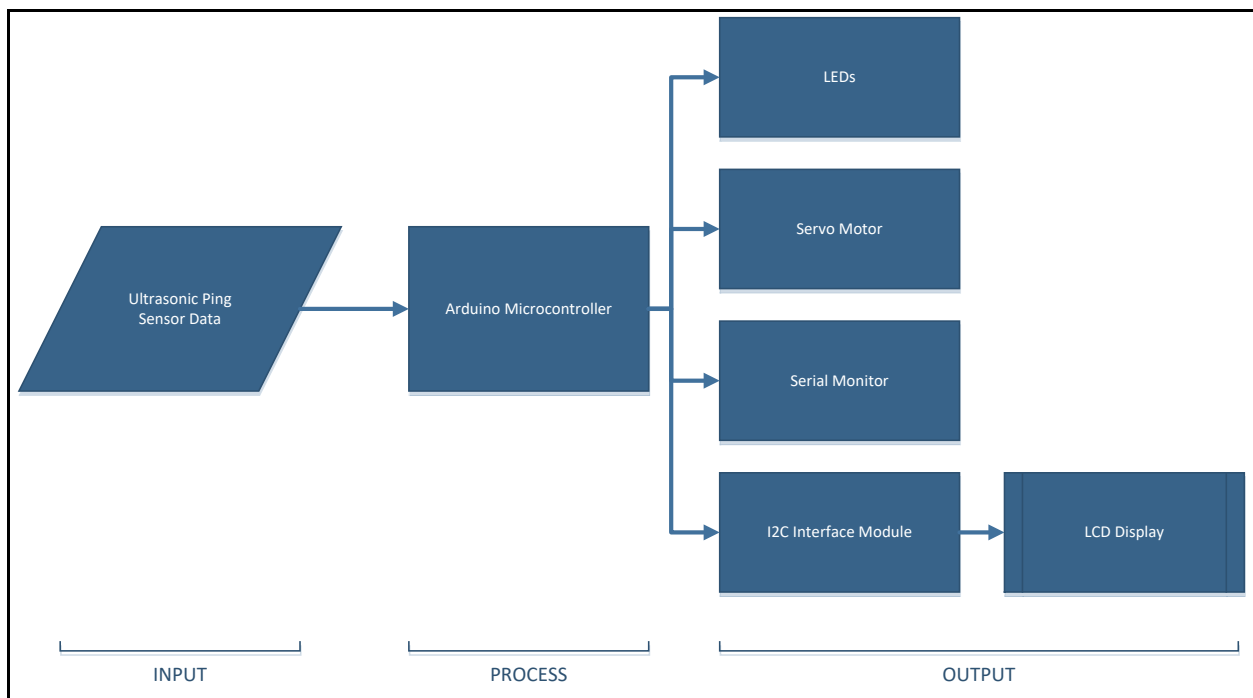


Figure 57: Block Diagram of the first input

The block diagram shows that the system is divided into three phrases. As far as the very first phrase is concerned, it collects data from the environment utilizing ultrasonic ping sensors. The data achieved from the sensor is utilized as input to the Arduino microcontroller. Later arduino microcontroller is programmed in order to take decision based on the data from the input & control the actuators. In this case the actuators are controlled output devices. The block diagram shows that there are four devices connected to the Arduino microcontroller as output. It includes

2 LED's, couple of servo motors, serial monitor & an LCD connected with I2C interface Module. The entire explained below is based on this block diagram.

ULTRASONIC PING SENSOR CODE

The very first thing is to add the libraries in the Arduino for the input & output components. A number of libraries come installed with the IDE but different libraries can be downloaded or created. In order to utilize an existing library just go to Sketch menu, select 'import library' & choose from the libraries accessible. This will embed a **#include** proclamation at the top of the sketch for every header (.h) file in the library's envelop. Such statements make the public functions defined by the library accessible/available for user's sketch. In addition they also signal the Arduino microcontroller in order to interface that library's code with user's sketch once it's compiled. As far as the proposed system coding is concerned, libraries available for the I2C serial Interface Module is utilized for the LCD and the servo motors.

```
#include <Wire.h> // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
```

Figure 58: Code Block 1

(Wire.h) library is basically for Arduino connection, (LiquidCrystal_I2C.h) was not available so it was downloaded from internet and then later added in Arduino library in order to define I2C module, (Servo.h) is actually a servo library where servo related codes are found.

```
//Defining the pin numbers
const int trigPinIn = 2;
const int echoPinIn = 3;

const int ledRout = 11;
const int ledGin = 10;

const int trigPinOut = 4;
const int echoPinOut = 5;
```

Figure 59: Code Block 2

As it can be seen in the figure illustrated above that we have created variables that are representing different pin numbers for input & output devices. Pin numbers are being stored in

different variables for defining the pins are constant integers. Using the predefined pin number values from the code, hardware has been built & implemented accordingly. Hence hardware is using the same connected pin number values. In figure above, trigger Pin for incoming & Outgoing are added in order to inform the microcontroller Arduino that Trigger Pins are connected to pins defined above. In the same manner echo Pin & LED's are also added.

```
// defines variables for in coming sensor
long durationIn;
int distanceIn;

// defines variables for out going sensor
long durationOut;
int distanceOut;
```

Figure 60: Code Block 3

It can be seen in the illustrated figure that the variables for the duration & distance for both ultrasonic ping sensors are defined. Basically these variables are utilized in order to store the time it takes for the sound wave to bounce & come back of an object. In others words, these variables are utilized to store the distance of the vehicle.

```
//Defining the servos
Servo servoIn;
Servo servoOut;

//Setting up the LCD with I2C module
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C address

//Define variables to be used
int slots = 6;
int incrementIN = 1;
int incrementOUT = 1;
```

Figure 61: Code Block 4

In figure above two servos are defined as (servoIn) & (servoOut). (servoIn) is for entrance gate & (servoOut) is for exit gate. The third line of the code in the above figure represents the setup of LCD utilizing I2C interface module. The LCD's address is supposed to be found by utilizing a port scanner and that's how it was done. The address was found to be placed in 0x3F address. LCD address is basically the address on which it communicates with LCD. As far as other numbers are concerned, they represent different pins of the LCD. At the end of third line the

term POSITIVE is utilized in order to turn the backlight on for the LCD. If POSITIVE is replaced by NEGATIVE then the LCD will work but the backlight will not turn on.

Later no of slots are identified that actually represent the no of slots available for vehicles in the parking system. The number of slots is assumed to be 6 as the prototype designed had a parking space for 6 vehicles only. 'incrementIN' & 'incrementOUT' are defined for the incrementing & decrementing purpose. They are defined as '1' so it only increment or decrement once for each vehicle passing.

```
void setup()
{
  Serial.begin(9600); // Used to type in characters

  lcd.begin(16,2); // initialize the lcd for 16 chars 2 lines, turn on backlight

  // ----- Quick 3 blinks of backlight -----
  for(int i = 0; i < 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
}
```

Figure 62: Code Block 5

Void setup () carry the set of code that is executed when the Arduino microcontroller is supplied with power. In addition it should be kept in mind that this code is only executed once. Moving forward to the first line of the code where baud rate is assigned as 9600. This is done so that the computer can communicate with Arduino Microcontroller by utilizing USB cable. The baud rate which is 9600 actually makes sure that the data from Arduino can be seen in the serial monitor (output). In the very next line lcd.begin is utilized in order to turn on the LCD display by using I2C interface module. In the very next line for loop is used where an auxiliary index variable is defined called 'i' that is denoted in for loop & is either incremented or decremented at the end of the loop. The index variable is assigned with the value of zero. As far as the value is between 0 & 3, the loop will repeat. At the end of the condition, the 'i' value is incremented that simply means that this set of code that is written within for statement can be executed for 3 times only. Lcd.backlight() and lcd.nobacklight() are utilized to turn on and to turn off the backlight for LCD followed with the command of delay(250) that is basically used to suspend execution of a program for a particular time in milliseconds. In addition this makes sure that LCD's backlight is turned ON for 0.25 seconds & then turned off for the same duration of time. The entire process will be repeated thrice.

```

lcd.setCursor (0,0);
lcd.print("Welcome To");
delay(250);
lcd.setCursor (0,1); // go to start of 2nd line
lcd.print("SMPS");
delay(1000);

}
lcd.backlight(); // finish with backlight on

```

Figure 63: Code Block 6

LCD that has been used is 16x2 that means that it consist of 16 columns & 2 rows. Lcd.setCursor is utilized to set the starting point of the text to be displayed on LCD screen following by the proposed topic information in the form of a string while completing the same information at the second row.

```

pinMode(6,OUTPUT);
servoIn.attach(6);
servoIn.write(0);

pinMode(7,OUTPUT);
servoOut.attach(7);
servoOut.write(0);

pinMode(ledRout, OUTPUT);|
pinMode(ledGin, OUTPUT);
pinMode(trigPinOut, OUTPUT);
pinMode(trigPinIn, OUTPUT);
}

```

Figure 64: Code Block 7

This is rest part of void setup() code. It can be observed by looking at the code that servoIn that is basically for the vehicles coming inside to park the vehicle is connected to pin no 6 whereas the other servo motor servoOut that is basically for the vehicles leaving the parking is connected to pin no 7. servoIn.write and servoOut.write is utilized in order to set their angles to zero degree which is done so that the servo motors initially remain in their normal position. In the last four lines, LED's & trigger pins are set to OUTPUT for ultrasonic ping sensors that basically helps the processor to send output signals by utilizing these pins.

```

void loop()
{
    // Clears the trigPin
    digitalWrite(trigPinIn, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPinIn, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPinIn, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    durationIn = pulseIn(echoPinIn, HIGH);
    // Calculating the distance in cm
    distanceIn= durationIn*0.034/2;
    delay(50);
}

```

Figure 65: Code Block 8

Void loop () is the main loop of the program in Arduino interface. In other words, the loop function is the main body of a loop that is running in a master program. The code written under void loop () represents the working of ultrasonic ping sensors (input) & is repeated until the setup loop is executed completely. ‘digitalWrite’ command is utilized to assign the value of the trigger pin (input) to LOW that basically clears the trigger pin & make sure that it is not sending any sound waves. Then a delay of 2 microseconds is added by utilizing ‘delayMicroseconds’. A delay of 2 microseconds is given so that the trigger pin can completely clear. ‘digitalWrite’ command is utilized again to assign the value of the trigger pin as HIGH for 10 microseconds. This means that during this time the trigger of the ultrasonic pin sensor is sending sound waves that are reflected back to the echo sensor after it bounces an object. After that, it is again turned off & the echo pin is set as HIGH that simply means that it is turned ON so that the sensor can calculate the time taken for the sound to wave go back & forth. distanceIn is the variable utilized to store the distance of the vehicle that is incoming which is done by utilizing mathematical equation to get the distance in centimetres as we know that speed of sound in air. Mathematical equation is given as follows:

$$S=vt$$

In addition it should be kept in mind that at the end of each sensor, the process of calculating the distance, a delay of 50 microseconds is kept in order to make sure that both the ultrasonic ping sensors doesn’t know interface with each other during operation. The process explained above is for incoming sensor & now the very same process will be repeated for the outgoing sensor.

```

Serial.print("distanceIn: ");
Serial.println(distanceIn);

Serial.print("distanceOut: ");
Serial.println(distanceOut);

```

Figure 66: Code Block 9

It can be seen in the figure above that Serial.print is used. Serial.print is used to display the distance of the incoming & outgoing vehicles on the serial monitor. Serial monitor is also an output that basically displays the distance data collected from the sensor along with the no of slots that are available on the LCD screen as LCD screen is not enough and it will not be able to display both data. Starting with the first line of the code 'distanceIn: '. It should be noted that there is a space in the string value in order to add the variable distanceIn next to it. Command is given to print the 'distanceIn: ' on the serial monitor. The very same process is then repeated for the distance for the outgoing vehicles.

```
if (distanceIn < 15 && incrementIN == 1 && slots > 0) //in coming sensor
{
  slots = --slots;
  servoIn.write(0); // Open servo gate
  digitalWrite(ledGin, HIGH);
  incrementIN = 0;
}
else if (distanceIn >= 15 && incrementIN == 0) { incrementIN = 1;
delay(500);
servoIn.write(80);
  digitalWrite(ledGin, LOW);
}
if (distanceOut < 15 && incrementOUT == 1 && slots < 6) //out going sensor { ++slots;
servoOut.write(0); // Open servo gate
  digitalWrite(ledGout, HIGH);
  digitalWrite(ledRout, LOW);
  incrementOUT = 0;
}
else if
(distanceOut >= 15 && incrementOUT == 0)
{ incrementOUT = 1;
delay(500);
servoOut.write(80);
  digitalWrite(ledGout, LOW);
  digitalWrite(ledRout, HIGH);
}
```

Figure 67: Code Block 10

The code written in the above illustrated figure is the most important code of the development process. If-else statement is utilized in order to make conditions so that the system's output work based on the input. If-else statement is using conditions which is that if the distance of the vehicle is less than 15cm then only slots will be decremented by utilizing '- - slots'. In addition 'slots >0' condition is given which means that if the number parking slots available are more than '0' then on this a signal will be sent to the servo motor and the barrier will be pulled up so that the vehicle can pass. At the very same time the green LED is turned ON that basically gives an indication that the vehicle can pass and as soon as the vehicle has passed barrier comes back to its original position making the LED light off. Green LED is turned on as 'HIGH' command is

given indicating that the vehicle is now allowed to pass. It is to be noted that there is a second condition in the if statement; the *incrementIN* must be 1. It can be seen in the above figure that the incrementIN was initially set as 1 while declaring the variables which means that when the very first vehicle arrives all the conditions of if-else statement are satisfied & the code is executed. At the end, incrementIN is set as zero which is done in order to make sure that the no of slots to be decremented only once when the vehicle is in range.

Another if-else statement is utilized for the distance greater than 15cm. Once gain incrementIn variable is set as 1 which actually makes sure that once the vehicle moves on, first if statement conditions are met when another vehicle comes nearby. Moreover a delay has been kept in order to make sure that the driver has some time to pass through before the barrier comes back to its original position & then at the end servo motor comes to its normal 'closed' position. Green LED is turned off as 'LOW' command is given.

The very same logic is utilized for exit gate. When the sensor senses the vehicle that is within the range (15cm) this means first condition is satisfied. Another condition is applied by using 'slots<6' which means that if the no of slots available in the parking lot is less than 6 then only servo will move 80° pulling the barrier or opening the gate and Green LED will turn on and as soon as the vehicle has passed barrier will come back to its original position making LED off. The condition of 0 and 6 applied for entrance and exit in order to make the system work in a way then only if the of empty slots are available then only vehicle can go inside in order to park his/her vehicle. Then an 'else statement' is used which shows if the sensor gets the distance more than 15cm then the servo motor doesn't move so as a result no increment is shown.

```
lcd.clear();  
lcd.setCursor(0,0); // go to start of 1st line  
lcd.print("No. of Lots: ");  
lcd.setCursor(0,1); // go to start of 2nd line  
lcd.print(slots);  
}
```

Figure 68: Code Block 11

This is the last block of code that is basically related to the displaying the no of slots in the parking space on LCD screen. Firstly, LCD is cleared from inputs that were displayed previously during the setup loop and then the cursor is set to the 1st row & 1st column. As we know that the LCD has 2 rows so the very first row is assigned to display the 'No of lots' whereas the 2nd row

is assigned to display the value of the variable called 'slots'. This variable is storing the no of slots available so the very same value is displayed on LCD screen so the driver can see.

Note: The entire program has been attached in the Appendix for reference.

PHOTOELECTRIC IR SENSOR FOR PARKING AID

Further moving to smart parking system that takes its other input from the Photoelectric IR sensor. This sensor is kept in order to sense the distance of the vehicle. The main idea is that 6 different Photoelectric IR sensors are placed for each parking lot. These sensors are actually fixed with the wall so that when the driver is parking his/her vehicle, distance can be measured and once the vehicle is in danger zone, the output of CFL bulb switches ON in order to alert the driver so that the driver can stop. The CFL bulb stays OFF until the vehicle is in safe zone.

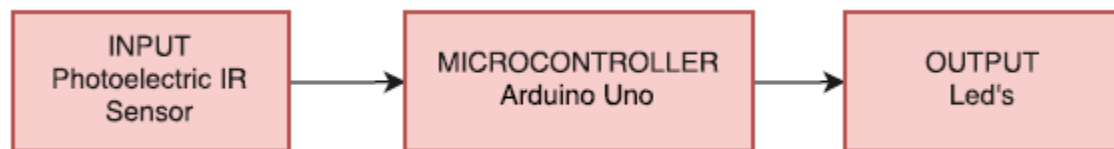


Figure 69: Block diagram for the 2nd input

The block diagram illustrated above shows the simple connection of Photoelectric IR sensor. In order to make the system run a block of code has to be written. The code to make the photoelectric IR sensor work is explained below:

PHOTOELECTRIC IR SENSOR CODE

Arduino libraries are basically the files written in C or C++ that provide your sketches with extra functionality for example the capability to control an LED matrix.

```
#include <Wire.h> // Comes with Arduino IDE
```

Figure 70: Code Block 11

The Arduino environment is extended with the utilization of libraries. They provide extra functionality for use specially working with hardware or manipulating data. As it can be seen in the figure illustrated above that (Wire.h) library is used for Arduino connection.

```

void setup() {
  pinMode(A0, INPUT);    //Receive signal from sensor
  pinMode(A1, OUTPUT);  //Ground pin
  pinMode(A2, OUTPUT);  //Vcc pin
  pinMode(11, OUTPUT);  //LED pin
  digitalWrite(A2, HIGH); //supply 5V to the terminal of sensor
  digitalWrite(A1, LOW);  // Connect sensor GND with Arduino Ground
  Serial.begin(9600);
}

```

Figure 71: Code Block 12

Void setup () function is called when the code is executed when the Arduino microcontroller is supplied with power. It is used to initialize variables, pin modes, start utilizing libraries, etc. Void setup () will run only one after every time Arduino board restarts. As it can be seen in the figure illustrated above that A1(ground pin) and A2 (Vcc Pin) are set to OUTPUT whereas A0 is set to INPUT so A0 will be used to read what photoelectric IR sensor is saying. Pin 11 is basically connected to LED which is basically the output that will be used as an indication to alert driver once the driver enters the danger zone. Then the photoelectric IR sensor is turned ON by using digitalWrite and making 'A2' HIGH which means 5V is supplied to the terminal of sensor by Arduino microcontroller whereas A1 is LOW which means that sensor ground is connected with Arduino ground. . Moving forward to the first line of the code where baud rate is assigned as 9600. This is done so that the computer can communicate with Arduino Microcontroller by utilizing USB cable. The baud rate which is 9600 actually makes sure that the data from Arduino can be seen in the serial monitor.

```

void loop() {
  Serial.println(analogRead(A0));  %Print voltage of output pin resolution (1-1024)
  delay(500);
  if(analogRead(A0) <500)
    digitalWrite(11, HIGH);
  else
    digitalWrite(11, LOW);
}

```

Figure 72: Code Block 13

Void loop () is the main loop of the program in Arduino interface. In other words, the loop function is the main body of a loop that is running in a master program. The code written under void loop () represents the working of Photoelectric IR sensors (input) & is repeated until the

setup loop is executed completely. The loop function is repeated continuously as it is called by a hidden loop that actually controls the execution of the Arduino Program. 'Serial.printIn' means print the data to the serial port. Data is basically the information achieved by the photoelectric sensor. So this will be printing the voltage of output pin resolution (1-1024). Then a delay of 0.50 microseconds is given by utilizing 'delay'. If-else statement is utilized in order to make conditions so that the system's output (LED) work based on the input. The code written within if-else statement is the most important code of the development code. If-else statement is utilizing conditions which is that if the reading achieved from input (IR sensor) is less than 500 as the reading can be anything from 1-1024, pin 11 is made HIGH which means that LED will turn ON followed by else statement that means that if the distance is greater than 500 then LED will remain OFF. If-else statement is used for the implementation of parking aid. The entire code can be summed up by saying that 500 is kept as safety point. When the distance of the vehicle is less than 500 from the wall then the red LED will switch ON indicating that the driver needs to stop as he/she can end up hitting his/her vehicle & when the distance of the vehicle is more than 500 from the wall, LED will remain OFF.

Note: The entire program has been attached in the Appendix for reference

Discussion- Program Development

In this assignment a smart parking system prototype model is designed, programmed and then implemented. The designed smart parking system is basically capable of showing the number of vehicles in the parking lot based on the incoming & outgoing vehicles. The problem statement mentioned above has been solved & the proposed design is implemented. In order to make the system smart, only 6 parking slots are designed for parking purpose. Then the program development is done in such a manner that as only 6 slots are available so only 6 vehicles can enter the parking lot which is quite unique. In any particular area or parking lots, this happens very commonly that plenty of vehicles are going around for the purpose of finding parking that results in increase of fuel consumption, higher energy wastage & more congestion. According to a study in the U.S the average distance drove to find a parking was a half-mile. Converting that to year it means 47000 gallons of petrol wasted. In order to avoid such kind of situation program development that has been explained block by block is done in a way that if there are parking slots available then only barrier will be pulled up or gate will open otherwise barrier will stay at its normal position. This system actually involves the use of sensor as an input to sense the data about the environment & then sends this data to the microcontroller that actually makes a decision based on the programmed logic & then sends an output signal to an output device to take that decision. According to the concept of project, the input sensor (ultrasonic pin sensor) actually collects data of distance that has been detected & based on the input the system works. When input is collecting data, 2 major cases were set in the programming part. As far as first case is concerned, the ultrasonic ping sensors detects vehicle further than 15cm whereas in the 2nd case, ultrasonic pin sensor detects object that is within the range of 15 cm. In both the cases system will act as follows:

1. First case is when the system's input is higher than 15cm:

In this case, when the system's input is higher than 15cm, the system will remain with no changes. As a result barrier will stay at this normal position & the results on LCD screen will not change.

2. Second case is when the system's input is less than 15cm:

In this case, when the system's input is receiving less than 15cm, the LED will turn on for the gate. The servo motor will open the barrier gate and the LCD will show a decrement of 1 if it is an incoming car, while if the vehicle is exiting LCD screen will show an increment of 1.

There are several problems that were encountered during the implementation. The problems were regarding software as well as the hardware. During initial testing, it was noticed that utilizing normal LCD without interface module will use too many pins on the Arduino Uno. This problem was solved by buying another LCD that comes with I2C interface module. I2C interface module limited the connection for the LCD to 4 pins only. The other problem faced was when LCD was connected but still results couldn't be displayed. The contrast of the LCD module is controlled by a potentiometer that needs to be switched on with the help of a screw in order to allow the LCD to correctly control the contrast. During early trials, when all components were directly connected with microcontroller Arduino, all the components drew current from the Arduino that actually caused problem in the circuit as noise was generated because of the servo sweep motion. As this noise was very random so the circuit didn't stop altogether but malfunctioned many times. In order to solve this problem an external power supply was used for connecting the servo motors with decoupling capacitors in order to reduce high frequency noise that is generated.

Another problem faced was during the setup of LCD utilizing I2C interface module. The LCD address that was found on internet was not setting this LCD. This problem was solved by little bit of research. It was found that the LCD address is supposed to be found by utilizing a port scanner & then that address is to be placed. LCD address is basically the address on which it communicates with LCD. LCD will not work until correct address is entered.

In addition when sensors were programmed, they worked correctly but sometimes the readings that were displayed were not accurate. The result shown was zero sometimes, the logic for the operation of the barrier is controlled by the servo motor, LED's & the number of slots available was controlled by the distance from the sensor which caused issues. After going through the code very thoroughly it was found that as both the sensors are employed in the model sent & received sound pulses without giving any delay that would actually cause interference among sensors &

as a result wrong readings will be achieved. This problem was then solved by using a short delay after the operation of each ultrasonic ping sensor.

As far as the other input of the system is concerned that is Photoelectric IR sensor. There are some situations in which the parking lots that are designed are very congested & might be very difficult to park in. It is very difficult for some drivers specially learners to park in these situations. In addition the rear distance between a wall or any other car that is parked is not visible that can actually cause accidents or the driver may end up bumping his car. Similar to the earlier concept another microcontroller is used with photoelectric IR sensor and LED. The main idea was to design and implement a system that can read the accurate distance between the vehicle & the wall or any car parked behind. As soon as the vehicle enters the danger zone (too close to the specific object like car or wall) LED will switch that is basically an alert alarm. The problem faced during implementation was that there were 6 parking lots so 6 different photoelectric IR sensors are supposed to be connected for each parking lot so as a result only one photoelectric sensor was used for implementation purpose. When input is collecting data, 2 major cases were set in the programming part. As far as first case is concerned, the Photoelectric IR sensor detects vehicle further than 500 whereas in the 2nd case, Photoelectric IR sensor detects object that is within the range of 500. In both the cases system will act as follows:

1. First case is when the system's input is higher than 500:

In this case, when the system's input is higher than 500, the system will remain with no changes & LED light this for alarming purpose will remain off.

2. Second case is when the system's input is less than 500:

In this case, when the system's input is receiving less than 500, the LED will turn on in order to alert the driver that he is danger zone & he/she is supposed to stop.

It is designed in order to help drivers & aid them in parking as it appears that many drivers are weak in parking section & as a result results can be seen on the car. This controller action that has been fitted in the wall will sense the vehicle arriving close & gives an alert indication with the help of a light to see. These LED lights can be replaced by alarms.

The prototype model is actually a complete model for parking system and with some improvements it can be installed in any parking area in order to display the no of car parks available. In order to improve the system, this circuit can be translated onto a PCB and as a result a more durable model for the circuit that can be connected with a relay in order to control an actual motor of a barrier. This would surely help to scale the model on a larger scale. This system has a limitation which is that when people passing by the sensor, it is considered as noise that would cause the sensor to think a vehicle is nearby & as a result barrier will be pulled up. In order to resolve this problem a metal sensor can be placed as well that will be detecting the metals so when people are passing by barrier will not open.

The last problem faced was when making the connections utilizing jumper wires. Even though the codes & connection were running without any error still LED's were not seem to be working. LED's were replaced but nothing seemed to work. The mistake was later identified that was when the jumper wires were cut into smaller lengths, upon the peeling, the wire got damaged. That damage didn't allow the conductivity of current through to reach its destination. Therefore extra care is required when cutting & peeling through a new batch of jumper wires.

There is always a room for improvement. Improvement can be done in hardware as well the software. The inputs and outputs are connected with proper methodological approach & program development was done by keeps all the aspects in mind of input and output components as well. All the errors in the code & issues in the hardware are sorted out & the yet existing limitations have been explained. Improvements required for the system were also stated along with possible method of approach.

output). Starting with the bulk capacitor of 100uF is connected to the voltage supply from the Arduino microcontroller to act as a decoupling capacitor & provide charge to the components when noise is present. A decoupling capacitor is a capacitor that is used to decouple one part of an electrical circuit from another. Noise that has been caused by other circuit elements is shunted through the capacitor, reducing the effect it has on the rest of the circuit. In addition it should be kept in mind that decoupling capacitors are commonly utilized in pairs. When a bulk capacitor of large value like 100uF is connected near the power supply then a small capacitor of lower capacitance is connected near the loads & that is called local capacitor. In order to do that couple of small capacitors of value .22 uF is utilized for the servo motors & couple of capacitors of value 10uF is utilized for the voltage regulator. As it can be seen in the figure illustrated above that the servo motors are powered externally & the voltage of the battery supply for the servo motor is 9V but as the servo motor operates at 5V so as a result a fixed voltage regulator is utilized in order to reduce the voltage to 5V. The reason for powering the servo motor externally is because during the servo motor sweep, it draws a lot of current which causes noise in the circuit. Hence, the supply line for the servo motor is separated from other components.

Note: Servo Motor SG90 datasheet is given in the output sections which shows that the servo motor used is of 5V so as a result 5V voltage regulator is used.

Looking at the connections, the servo motor exit and entry are connected to digital pins 7 and 6 respectively at Arduino whereas its drawing current from the external battery. The digital pins send the pulse to servo motor for movement. Next the lcd pins SDL and SDA are connected to A5 and A4 analog pins of the microcontroller, to give data to display and they are clocked. Further the Vcc and Ground of the Lcd is again connected to external 5V coming from a battery of 9V and it is controlled using a voltage regulator. The bulbs are connected on either side of the lot and are made real time by connecting a relay to them. These relays draw power from the external battery source and are connected to the bulb at the other end. The IN pin of the relay at exit and entry are connected to 10 and 11 digital pin of the controller which will give the trigger to be high or be low so as to turn the bulb on or off. Both the relays are then connected to the bulbs whose one wire is connected from power supply to the bulb holder while the other one wire of the power supply cable comes to NC port of relay and the bulb holders wire come to the common port. The task of the relay here is when the trigger pin is high given in from the

controller it will connected the switch and the bulb will be on, while when its low the switch is kept off and connection to the bulb is not there. Finally the Arduino can be powered with the help of a laptop USB port or a battery supply can be taken from the breadboard.

List of components utilized is tabulated below:

No.	Component	Quantity
1	Arduino Uno Rev 3	1
2	Ultrasonic Ping Sensor HC-SR04	2
3	CFL bulb (18W)	2
4	Servo Motor 90G	2
5	LCD 16x2 Display	1
6	I2C Serial Interface Module	1
7	9V battery supply	2
8	Relay board with Transistor (JQC-3F-S-Z, 5Vdc)	2
9	5V voltage regulator (LM7805)	2
10	Polarized capacitor 100uF	1
11	Polarized capacitor .22uF	2
12	Polarized capacitor 10uF	2
13	Bread board	1
14	Jumper wires	NA

The figure below illustrates the connection of smart parking which is the actual implementation in the prototype model

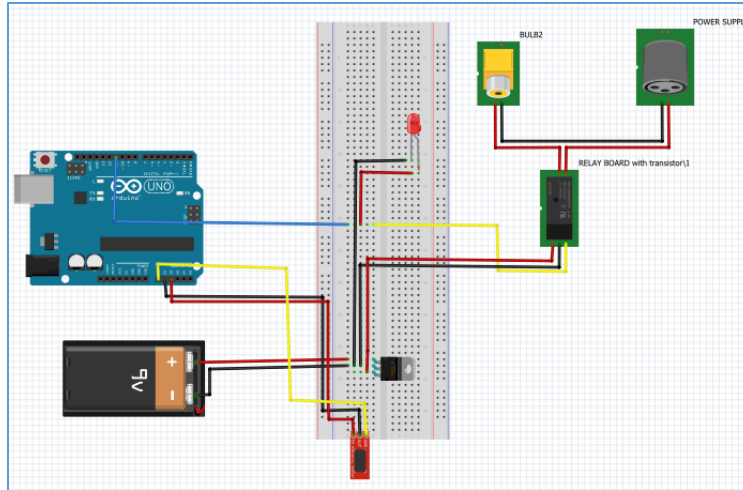


Figure 74: Complete Circuit of smart parking

The input and output connections have been explained earlier in this report. As far as the illustrated figure is concerned, it represents the collective connection of the components (input & output). The connection is quite simple but the results achieved are quite impressive & useful for real time applications. Starting with Photoelectric IR sensor that has 3 pins (Vcc, ground & OUT) which are connected with Arduino microcontroller analog pins.

Photoelectric IR sensor Ground OUT pin is connected to A0

Photoelectric IR sensor Ground pin is connected to A1

Photoelectric IR sensor Vcc pin is connected to A2

A0 will be used to read what photoelectric IR sensor is saying. This pin will basically give the distance to the IR sensor and then the signals will be sent to the output LED which is connected to pin 11 of Arduino board. Code has already been explained under Program development. Pin 11 is basically connected to LED which is basically the output that will be used as in indication to alert driver once the driver enters the danger zone.

The real time application was also implemented by just connected another relay board, figure below shows the connection

List of components utilized is tabulated below:

No.	Component	Quantity
1	Arduino Uno Rev 3	1
2	Photoelectric IR sensor	1
3	Red LED	1
4	Bread board	1
5	Relay board with Transistor (JQC-3F-S-Z, 5Vdc)	1
6	CFL bulb (18W)	1
7	Jumper wires	NA

Innovation and Enhancement

No matter what the system is based on. Whatever and however perfectly designed a control system is, there is always a room for improvement & innovation. In the world today, technology is advancing at a very rapid pace, the system needs to grow and improve accordingly. Similarly, for the current smart parking system some enhancement or innovations can be done. If there are different floors for parking categorized into VIP parking, visitor parking etc. In this case, the member who is supposed to park in VIP parking must have a code. When he enters via mobile phone the frequency generated by the keys will be decoded into their BCD format by the DTMF decoder that will be fed to CPLD & thus the CPLD will send the commands to the motor driver to open the gate so that vehicle can enter the VIP slot. This improvement is recommended for the parking areas that are divided into different categories like VIP, visitors parking etc. This is how the system will be providing the VIP parking slot to the driver that has entered the correct password. This can be implemented on a larger scale. This innovation will help drivers who are not well aware of parking so they end up parking at some ones else place and get their car tow. As far as the current system is concerned, for the second input (photoelectric IR sensor), the

output that is basically an LED light can be replaced by buzzer so the user can hear an alarm when in danger.

In addition the innovation that has been implemented is done within the coding. Normally smart parking system is designed in a way that if there are 100 slots available in the parking lot just an indication is given on LCD that 'parking is full' or something like that which may not be noticed by the driver. As far as implementation of this system is concerned, as there are 6 slots available so LCD will be showing that slots are full & with that gate will not open. An opening command will only be sent to the gate when there is an empty spot. As a result when gate will not open, driver will notice immediately that parking space is full. This will surely saving driver's time and fuel. As there is no sense of going inside the parking lot if there is no space available for parking.

Conclusion

All the objectives and tasks of the assignment were achieved swiftly and in a manner, that discovering new techniques and knowledge being one of the core target rather than just completing the task. The design and procedure of the project was done in accordance to the criteria given and necessary changes were made to get the desired optimum results. The challenges faced while completing this project were discussed and ample solutions were given to resolve the issues and progressive suggestions were made to make this project even more reliable and efficient in terms of future for example if this system had to be used in a private garage then a identification algorithm could be added so that only identified vehicles could enter only.

One of the things learnt from the experience of designing this system, is the importance of a microprocessor. A device that uses a microprocessor is normally capable of many functions, such as word processing, calculation, and communication via Internet or telephone. However, for the device to work properly, the microprocessor itself has to communicate with other parts of the device. For example, a microprocessor would need to communicate with the video display to control the output data that a program may produce. Therefore, a microprocessor would act as device's "brain" in that it transmits, receives and interprets the data needed to operate a device. Today's society would be nothing like it is now without the microprocessor. Most every electronic device has a microprocessor in it, from garage door openers, to remote controls, to cell phones and ipods etc

References

HC-SR04, U. (2015). Ultrasonic Sensor - HC-SR04 - SEN-13959 - SparkFun Electronics. [online] Sparkfun.com. Available at: <https://www.sparkfun.com/products/13959> [Accessed 2 Sep. 2017].

Gantelet, E. and Lefauconnier, A. (2006). THE TIME LOOKING FOR A PARKING SPACE: STRATEGIES, ASSOCIATED NUISANCES AND STAKES OF PARKING MANAGEMENT IN FRANCE. European Transport and contributors.

Walker, A. (2014). This New Parking App Can Find Empty Spaces, No Sensors Required. [online] Gizmodo.com. Available at: <http://gizmodo.com/this-new-parking-app-can-find-empty-spaces-no-sensors-1635449274> [Accessed 19 Aug. 2017].

Glatz, B. (2015). How Far Do We Really Drive While Looking for Parking?. [online] Fybr-tech.com. Available at: <http://www.fybr-tech.com/how-far-do-we-really-drive-while-looking-for-parking/> [Accessed 31 Aug. 2017].

Study.com. (n.d.). Input, Processing, Output & Feedback: Information System Components - Video & Lesson Transcript | Study.com. [online] Available at: <http://study.com/academy/lesson/input-processing-output-feedback-information-system-components.html> [Accessed 1 Sep. 2017].

www.tutorialspoint.com. (n.d.). System Analysis and Design Input / Output and Forms Design. [online] Available at: https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_input_output_forms.htm [Accessed 2 Sep. 2017].

W9xt.com. (n.d.). Microcontroller Digital Output Basics. [online] Available at: http://www.w9xt.com/page_microdesign_pt3_ouput_basics.html [Accessed 2 Sep. 2017].

Science Buddies. (n.d.). Introduction to Servo Motors. [online] Available at: <https://www.sciencebuddies.org/science-fair-projects/references/introduction-to-servo-motors> [Accessed 4 Sep. 2017].

Earl, B. (2015). Servo Motors | Adafruit Motor Selection Guide | Adafruit Learning System. [online] Learn.adafruit.com. Available at: <https://learn.adafruit.com/adafruit-motor-selection-guide/rc-servos> [Accessed 3 Sep. 2017].

Coker, E. (2016). Understanding Types of Servo Motors and How They Work | Make:. [online] Make: DIY Projects and Ideas for Makers. Available at: <http://makezine.com/2016/05/13/understanding-types-of-servo-motors-and-how-they-work/> [Accessed 3 Sep. 2017].

Electronics Hub. (2016). Arduino Servo Motor. [online] Available at: <http://www.electronicshub.org/arduino-servo-motor/> [Accessed 5 Sep. 2017].

Anon, (n.d.). SG90 9g Micro Servo. [online] Available at: <http://akizukidenshi.com/download/ds/towerpro/SG90.pdf> [Accessed 2 Sep. 2017].

WhatIs.com. (n.d.). What is light-emitting diode (LED)? - Definition from WhatIs.com. [online] Available at: <http://whatis.techtarget.com/definition/light-emitting-diode-LED> [Accessed 3 Sep. 2017].

Anon, (n.d.). CFLs vs LEDs. [online] Available at: <https://www.greenamerica.org/green-living/cfls-vs-leds-better-bulbs> [Accessed 5 Sep. 2017].

Anon, (2000). COMPACT FLUORESCENT LAMP. [online] Available at: <http://docs-europe.electrocomponents.com/webdocs/0e52/0900766b80e522ad.pdf> [Accessed 8 Sep. 2017].

Circuitar.com. (n.d.). LCD - 16x2 character LCD with I2C interface - Arduino-compatible shields - Circuitar. [online] Available at: <https://www.circuitar.com/nanoshields/modules/lcd/> [Accessed 1 Sep. 2017].

Datsi.fi.upm.es. (2008). arduino-info - LCD-Blue-I2C. [online] Available at: http://www.datsi.fi.upm.es/docencia/Informatica_Industrial/DMC/LCD-Blue-I2C.htm [Accessed 3 Sep. 2017].

Appendix

```
#include <Wire.h> // Comes with Arduino IDE
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

//Defining the pin numbers
const int trigPinIn = 2;
const int echoPinIn = 3;

const int ledRout = 11;
const int ledGin = 10;

const int trigPinOut = 4;
const int echoPinOut = 5;

// defines variables for in coming sensor
long durationIn;
int distanceIn;

// defines variables for out going sensor
long durationOut;
int distanceOut;

//Defining the servos
Servo servoIn;
Servo servoOut;

//Setting up the LCD with I2C module
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE); // Set the LCD I2C address

//Define variables to be used
int slots = 6;
int incrementIN = 1;
int incrementOUT = 1;

void setup()
{
  Serial.begin(9600); // Used to type in characters

  lcd.begin(16,2); // initialize the lcd for 16 chars 2 lines, turn on backlight
```



```

// ----- Quick 3 blinks of backlight -----
for(int i = 0; i < 3; i++)
{
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);

    lcd.setCursor (0,0); // go to start of 1st line
    lcd.print("Welcome To");
    lcd.setCursor (0,1); // go to start of 2nd line
    lcd.print("SMPS");
    delay(1000)

}
lcd.backlight(); // finish with backlight on

pinMode(6,OUTPUT);
servoIn.attach(6);
servoIn.write(0);

pinMode(7,OUTPUT);
servoOut.attach(7);
servoOut.write(0);

pinMode(ledRout, OUTPUT);
pinMode(ledGin, OUTPUT);
pinMode(trigPinOut, OUTPUT);
pinMode(trigPinIn, OUTPUT);
}

```

```

}

void loop()
{
    // Clears the trigPin
    digitalWrite(trigPinIn, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPinIn, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPinIn, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    durationIn = pulseIn(echoPinIn, HIGH);
    // Calculating the distance in cm
    distanceIn= durationIn*0.034/2;
    delay(50);

    // Clears the trigPin
    digitalWrite(trigPinOut, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPinOut, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPinOut, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    durationOut = pulseIn(echoPinOut, HIGH);
    // Calculating the distance
    distanceOut= durationOut*0.034/2;
    delay(50);

    Serial.print("distanceIn: ");
    Serial.println(distanceIn);

    Serial.print("distanceOut: ");
    Serial.println(distanceOut);

```

```

if (distanceIn < 15 && incrementIN == 1 && slots>0 ) //In coming sensor
{
    slots = --slots;
    servoIn.write(0); // Open servo gate
    digitalWrite(ledGin, HIGH);
    incrementIN = 0;
}
else if (distanceIn >= 15 && incrementIN == 0)
{
    incrementIN = 1;
    delay(500);
    servoIn.write(80);
    digitalWrite(ledGin, LOW);
}

if (distanceOut < 15 && incrementOUT == 1 && slots<6) //out going sensor
{
    servoOut.write(0); // Open servo gate
    digitalWrite(ledGout, HIGH);
    digitalWrite(ledRout, LOW);
    incrementOUT = 0;
}
else if (distanceOut >= 15 && incrementOUT == 0)
{
    incrementOUT = 1;
    delay(500);
    servoOut.write(80);
    digitalWrite(ledGout, LOW);
    digitalWrite(ledRout, HIGH);
}

lcd.clear();
lcd.setCursor (0,0); // go to start of 1st line
lcd.print("No. of Lots: ");
lcd.setCursor (0,1); // go to start of 2nd line
lcd.print(slots);
}

```