East West University

UNDERGRADUATE PROJECT REPORT

ON

SPARK: IOT Based Smart Parking System

Submitted By

Mashuk Kowser

2013-2-60-052

Shibly Sirajee

2013-2-60-035

Supervised By: Shakila Mahjabin Tonni

Lecturer, Department of Computer Science and Engineering,

East West University

A project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science and Engineering to the Department of Computer Science and Engineering.

August 10, 2017

DECLARATIONS

This is to certify that this project is an original work and was done by us and it has not been submitted elsewhere for the requirement of any other purposes.

Signature of the Students

…… **……..**

Mashuk Kowser **Shibly Sirajee**

ACCEPTANCE

This project is entitled "**SPARK: IOT Based Smart Parking System**" submitted by Mashuk Kowser, ID no. 2013-2-60-052 and Shibly Sirajee, ID no. 2013-2-60-035 to the Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh is accepted as satisfactory for partial fulfillment of the requirement for the degree of Bachelor of Computer Science and Engineering on August 2017.

.....

Shakila Mahjabin Tonni
Lecturer
Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh.

……

Dr. Mozammel Huq Azad Khan
Chairperson and Professor
Department of Computer Science and Engineering
East West University, Dhaka, Bangladesh.

ABSTRACT

Lack of availability of parking facilities for the increasing number of vehicles used daily creates chaos and results in impassable traffic jams. In contrast to the situation, often there are empty parking spaces. But limited accesses to proper information on space availability, people are forced to park vehicles on the roadside. This dreadful scenario can be made tolerable by simply providing information to the vehicle users about the nearest available parking spots, so that, they can board their vehicles in a place other than the busy road. In this project, we concentrate on finding a way to reduce off-street parking effectively. This is done by developing a smart way to find an available parking spot just by sensing the presence of a vehicle in an area. This project is built on IoT based architecture using Arduino Mega device along with some sensors and motors. The web application part is handled using PHP.

ACKNOWLEDGEMENT

First of all, we are grateful to Ms. Shakila Mahjabin Tonni, Lecturer, Department of Computer Science and Engineering, East West University for her precious guidance, consistent advice and support for the completion of this project. Without her support, it was not possible for us to complete this task.

We would like to express our gratitude to all the faculty members of the Department of Computer Science and Engineering for their encouragements and co-operations we found in the process.

But above everything, we like to thank Almighty Allah for giving us the ability, patience, and determination in completing the project successfully.

Table of Contents

Abbreviation and Acronyms

IoT: Internet of Things

GDP: Gross Domestic Product

IDE: Integrated Development Environment

LED: Light Emitting Diode

ICSP: In-Circuit Serial Programming

AC: Alternating Current

DC: Direct Current

USB: Universal Serial Bus

VIN: Voltage Input

EEPROM: Electrically Erasable Programmable Read Only Memory

TCP: Transmission Control Protocol

SPI: Serial Peripheral Interface

PWM: Pulse Width Modulator

LDR: Light Dependent Resistor

List of Figures

Chapter 1 Introduction

The number of vehicles we belong in our city that number of parking space we do not have actually. Due to the lack of parking space, during office hour or another time, we need park our cars on-street. So this on-street parking makes the roads narrow for other moving vehicles and causes a traffic jam. Sometimes car stealing or car damage incident occurs due to on-street parking as there is no way of securing cars on an open street. Parking car in a wrong place may cause police wrecking also. All the parking spaces exist into the building are not properly organized. We are actually working on it. Our system provides a systematic car parking solution with strong security. Any registered user will ask a parking space through a web application. Then they will be shown all the parking lots with available spaces. Whenever they park their car, they need to press the lock button on the web application to make sure the parking bar is locked. Until they press the unlock button providing their passcode, the Parking Bar will not unlock.

1.1 About IoT

Internet of Things (IoT) is a bury organizing framework which is utilized to send information or data through any info gadget to a disjoint and a particular basic leadership process or calculation is connected to control any gadget in view of that information or data. 'S.PARK: Smart Parking System' is a real world application of IoT. In an ‘S.PARK ' all the physical objects or devices are interconnected through a network and they use Login/Registration module, sensors, servo motor, registers etc. to collect valid user information, parking space information and show car information to the user based on search criteria. For example, if a user is not registered, he/she

must have to be registered. After getting registered, he/she can search the parking space provided by this system S.PARK. Based on the search criteria, a user can see the available parking space in any location. After parking his/her vehicle, he can lock the gate for safety and a secret number will be shown. While getting his/her vehicle back, the secret number will be required. This Smart Parking System will help to improve the resource utilization, reduce traffic jam. In our research, we will work on a small portion of an area which indicated a simple simulation.

1.2 Motivation

In Bangladesh, Traffic jam is one of the biggest problems in current times. There are so many reasons behind it, but Off-Street Parking is a major issue mostly in Dhaka city. As we are a developing nation, this traffic jam is pulling us behind against the sorghum of development. In our country, we are very poor in controlling off-street parking management, the government is aware of that, but the agent of the government, Police takes the bribe for that. So this problem is never solved. A study shows, traffic congestion in capital eats up 20,000 crore taka a year in Bangladesh. Some 32 lakh business hours are lost to this curse in every day [1]. These things make a huge negative impact in our countries GDP. So, by thinking this we become motivated to develop a system which will provide a smart, user-friendly and secure way to search and park in an available parking lot that saves our time and money.

1.3 Objectives

- Providing information on available parking spaces to the registered users.
- Automatically sensing vehicle's presence and update website information on a specific interval.
- Providing an efficient way to manage spaces for the parking facility providers.
- An automated and authorized gate locking system for each parking spot.

1.4 Overview of the proposed system

The main controlling component of the project is a microcontroller board design and a web based user interface. Arduino microcontroller board takes sensor inputs and display on the browser. If a car is placed on a parking spot, the servo motor goes down blocking the entrance if the spot by sensing LDR sensor signal, and generates a PIN number for unlocking (moving up the motor) the gate. If the user wishes to unlock the gate and take the vehicle away from the spot, he needs to give the previously generated PIN number. The servo motor initially keeps the angle to 0 degrees, meaning the parking bar is open. The motor moves from 0 to 90 only when the user enters the valid pin to unlock their car. There will be a caution sign that only registered user can park their car otherwise their car will be fined for unauthorized entry.

1.5 Contribution

Object sensing by the microcontroller is an already implemented work. But, we used it to create a web based information hub for the user. Also, we demonstrate how unauthorized access to parked vehicles can be prevented by using user authentication based gate locking system. In this work, we present Arduino Uno microcontroller interfacing with the computer by programming the instructions in the Arduino IDE. The web application is developed by PHP language and the database in MySQL database. We have used web command to handle the hardware system.

1.6 Outline of this paper

Chapter 1 provides an introduction to the Smart Parking System (S.Park).
Chapter 2 describes the hardware architecture of this project and necessary steps of interfacing.
Chapter 3 describes the working procedure of this project and necessary Arduino functions.
Chapter 4 evaluates the working system and,
Chapter 5 includes the concluding remark and suggestion on possible future improvements.

Chapter 2 Related Works

In this chapter, we describe the entire proposed system of our project where all the necessary components name, their significance in the project are described elaborately.

2.1 Overview of Arduino

Arduino is an open-source gadget stage in view of simple to-utilize equipment and programming. Arduino sheets can read inputs - light on a sensor, a finger on a catch, or a Twitter message - and transform it into a yield - enacting an engine, turning on a LED, distributing something on the web. You can guide your board by sending an arrangement of guidelines to the microcontroller on the board. To do as such you utilize the Arduino programming dialect (in view of Wiring), and the Arduino Software (IDE), in light of Processing.

Throughout the years Arduino has been the cerebrum of thousands of activities, from ordinary articles to complex logical instruments. An overall group of producers - understudies, specialists, craftsmen, developers, and experts - has accumulated around this open-source stage, their commitments have meant a mind blowing measure of available information that can be of extraordinary help to learners and specialists alike.

All Arduino sheets are totally open-source, engaging clients to assemble them autonomously and in the long run adjust them to their specific needs. The product, as well, is open-source, and it is becoming through the commitments of clients around the world. In our proposed framework, we have utilized Arduino Mega.[2]

2.2 Arduino Mega

The Arduino Mega is a microcontroller board in light of the ATmega1280. It has 54 advanced information/yield pins, 16 simple sources of info, 4 UARTs (equipment serial ports), a 16 MHz precious stone oscillator, a USB association, a power jack, an ICSP header, and a reset catch. It contains everything expected to help the microcontroller; basically, interface it to a PC with a USB link or power it with an AC-to-DC connector or battery to begin [3]

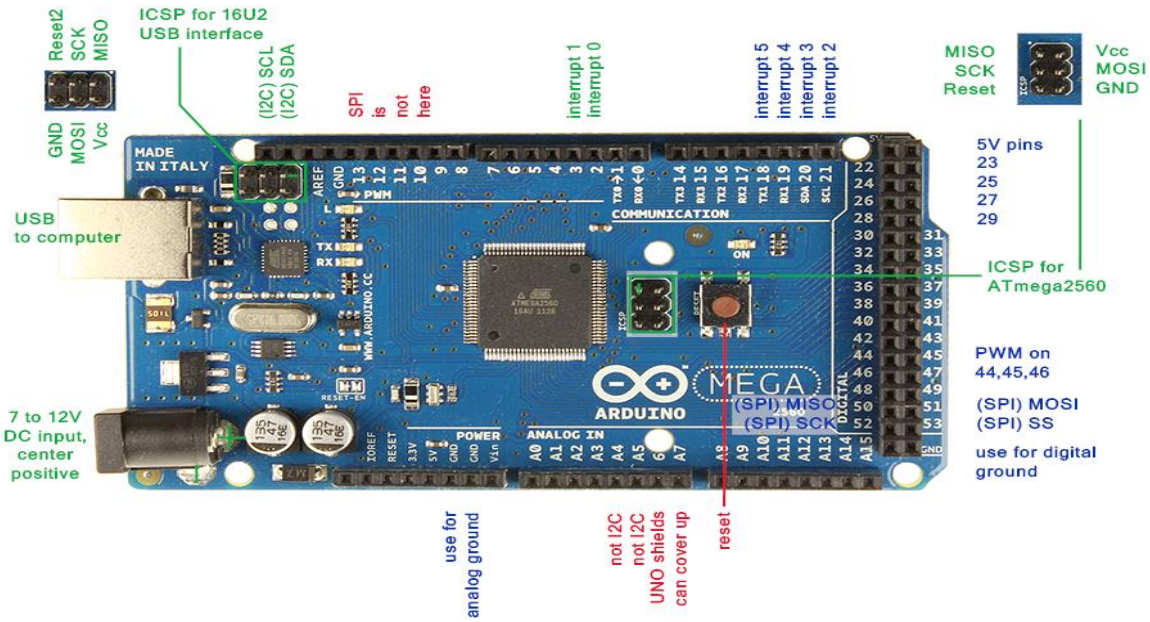

Figure 2.1: Arduino Mega

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Figure 2.2: Technical Specification [4]

2.3 Arduino Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and VIN pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts

2.4 Arduino Power PINS:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or another regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.**The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.
- **GND.** Ground pins.

2.5 Arduino Memory

The ATmega1280 has 128 KB of flash memory for storing code (of which 4 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

2.6 Arduino Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 K Ohms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL

serial data. Pins 0 and 1 are also connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.

- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM: 2 to 13 and 44 to 46.** Provide 8-bit PWM output with the analogWrite () function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove or Diecimila.

The Mega has 16 analog inputs, each of which provides 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference () function.

There are a couple of other pins on the board:

- **AREF.** The reference voltage for the analog inputs. Used with analogReference ().

- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

2.7 Arduino LEDs

Likewise, the Arduino has four LEDs: **L**, **RX**, **TX** and **ON.** On the Mega, **ON** PIN is on the bottom-right, **L, TX, RX** is on the Top-Right corner of Arduino mega.

ON LED: This LED will shine green whenever the Arduino is powered. Always check this LED if your Arduino is not acting right, if it's flickering or off then you should check your power supply.

RX and TX LEDs: These are like '**send'** and '**receive'** LEDs on your cable modem. They blink whenever information is sent from or not to the Arduino through the USB connection. The **TX** LED lights up yellow whenever data is sent from the Arduino to the computer USB port. The **RX** LED lights up yellow whenever data is sent to the Arduino from the computer USB port.

L LED: This is the one LED that you can control. The **ON**, **RX** and **TX** LEDs all light up automatically no matter what. The **L** LED, however, is connected to the Arduino main chip and you can turn it on or off when you start writing code and uploading on it.

2.8 Arduino Communication

The Arduino Mega has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega1280 provides four hardware UARTs for TTL (5V) serial communication. An FTDI FT232RL on the board channels one of these over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual

data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega1280 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify the use of the I2C bus.

2.9 Ethernet Shield

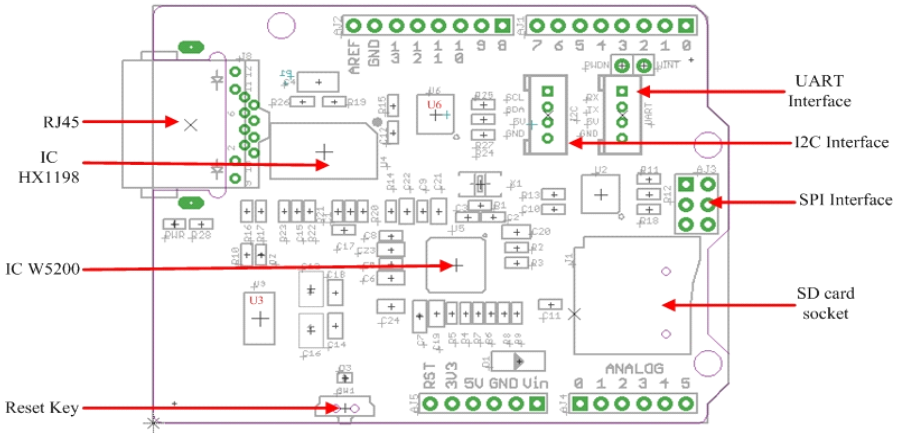

Figure 2.3: Ethernet Shield

The Arduino Ethernet Shield allows an Arduino board to connect to the internet. It is based on the Wiznet-W5100 Ethernet chip. The Wiznet-W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The Ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top. The latest

revision of the shield adds a micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Duemilanove and Mega (using the Ethernet library coming in Arduino 0019). An SD card library is not yet included in the standard Arduino distribution. The latest revision of the shield also includes a reset controller, to ensure that the W5100 Ethernet module is properly reset on power-up. Previous revisions of the shield were not compatible with the Mega and need to be manually reset after power-up. The original revision of the shield contained a full-size SD card slot; this is not supported. Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 11, 12, and 13 on the Duemilanove and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general I/O. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work. Note that because the W5100 and SD card shares the SPI bus, only one can be active at a time. If you are using both peripherals in your program, this should be taken care of by the corresponding libraries. If you're not using one of the peripherals in your program, however, you'll need to explicitly deselect it. To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output. The shield provides a standard RJ45 Ethernet jack.

The reset button on the shield resets both the W5100 and the Arduino board.

2.10 Ethernet Shield Features:

- ATMega328 Microcontroller.
- Operating Voltage: 5V.
- Input Voltage: 7-12V.
- Digital I/O Pins: 14 (4 of which are used by the Ethernet controller).

- Analog Input Pins: 6.
- On Board Micro SD Card slot.
- W5100 TCP/IP Embedded Ethernet Controller.

2.11 Ethernet Shield LEDs:

- **PWR**: Indicates that the board and shield are powered
- **LINK**: Indicates the presence of a network link and flashes when the shield transmits or receives data
- **FULL D**: Indicates that the network connection is full duplex
- **100M**: Indicates the presence of a 100 Mb/s network connection (as opposed to 10 Mb/s)
- **RX**: Flashes when the shield receives data
- **TX**: Flashes when the shield sends data
- **COLL**: Flashes when network collisions are detected.

The solder jumper marked "INT" can be connected to allow the Arduino board to receive interrupt-driven notification of events from the W5100, but this is not supported by the Ethernet library. The jumper connects the INT pin of the W5100 to digital pin 2 of the Arduino.

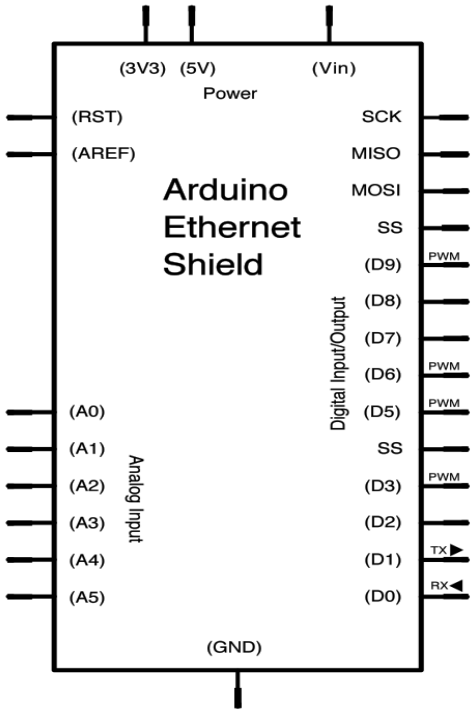

Figure 2.4: Schematic Diagram of Ethernet Shield

2.12 W5100 Datasheet

The W5100 is a full-featured, single-chip Internet-enabled 10/100 Ethernet controller Designed for embedded applications where ease of integration, stability, performance, area and system cost control are required. The W5100 has been designed to the ciliate easy implementation of Internet connectivity without OS. The W5100 is IEEE 802.3 10BASE-T and 802.3u 100BASE-TX compliant.

The W5100 includes fully hardwired, market-proven TCP/IP stack and integrated Ethernet MAC & PHY. Hardwired TCP/IP stack supports TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE which has been proven in various applications for several years. 16Kbytes internal buffer is included for data transmission. No need of consideration for handling Ethernet Controller, but

simple socket programming is required. For easy integration, three different interfaces like memory access way, called direct, indirect bus and SPI, are supported on the MCU side.[5]

2.13 Target Application for W5100

The W5100 is well suited for many embedded applications, including:

- Home Network Devices: Set-Top Boxes, PVRs, Digital Media Adapters
- Serial-to-Ethernet: Access Controls, LED displays, Wireless AP relays, etc.
- Parallel-to-Ethernet: POS/ Mini Printers, Copiers
- USB-to-Ethernet: Storage Devices, Network Printers
- GPIO-to-Ethernet: Home Network Sensors
- Security Systems: DVRs, Network Cameras, Kiosks
- Factory and Building Automation
- Medical Monitoring Equipment's
- Embedded Servers

2.14 W5100 Features

- Support Hardwired TCP/IP Protocols: TCP, UDP, ICMP, IPv4 ARP, IGMP, PPPoE, Ethernet
- 10BaseT/100BaseTX Ethernet PHY embedded
- Support Auto Negotiation (Full-duplex and half duplex)
- Support Auto MDI/MDIX
- Support ADSL connection (with support PPPoE Protocol with PAP/CHAP Authentication mode)
- Supports 4 independent sockets simultaneously

- Not support IP Fragmentation
- Internal 16Kbytes Memory for TX/RX Buffers
- 0.18 μm CMOS technology
- 3.3V operation with 5V I/O signal tolerance
- Small 80 Pin LQFP Package
- Lead-Free Package
- Support Serial Peripheral Interface(SPI MODE 0, 3)
- Multi-function LED outputs (TX, RX, Full/Half duplex, Collision, Link, Speed)

2.15 W5100 Block Diagram:

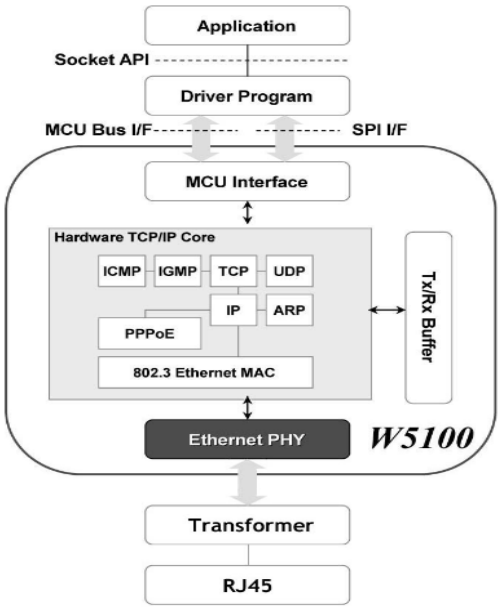

Figure 2.5 Block Diagram of W5100

2.16 W5100 Pin Assignment:

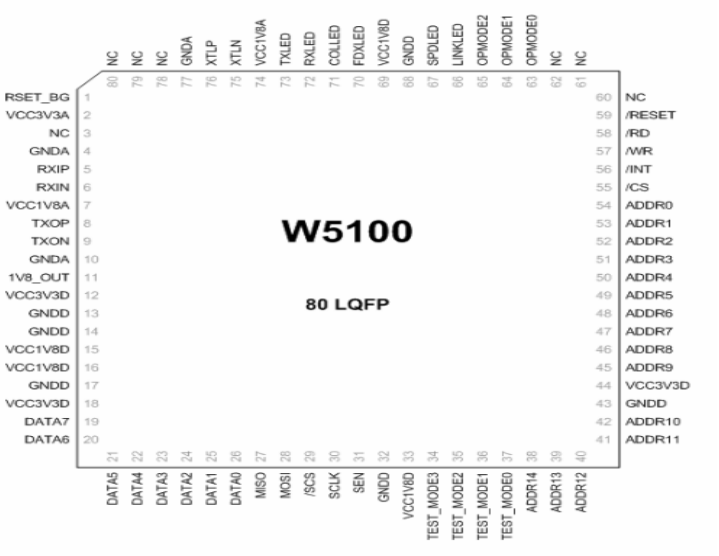

Figure 2.6 Pinout of W5100

2.17 W5100 Memory Mapping:

W5100 is composed of Common Register, Socket Register, TX Memory, and RX Memory as shown on the next page:

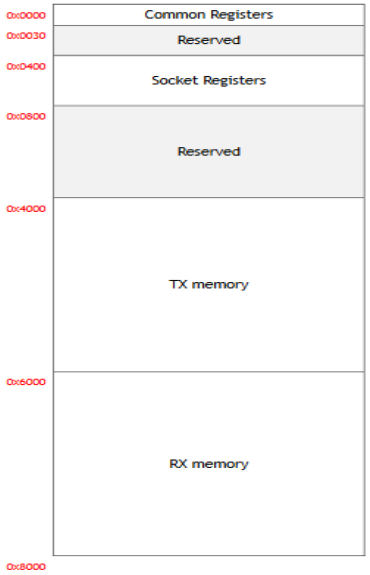

Figure 2.7: Memory Mapping W5100

2.18 LDR Sensor

Figure 2.8: LDR Sensor

It is a 5MM Photoresistor or Light Dependent Resistor (LDR). A Photoresistor (or light-dependent resistor, LDR, or photoconductive cell) is a light-controlled variable resistor. The resistance of a Photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A Photoresistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.

A Photoresistor is made of a high resistance semiconductor. In the dark, a Photoresistor can have a resistance as high as several mega ohms (MΩ), while in the light, a Photoresistor can have a resistance as low as a few hundred ohms. If incident light on a Photoresistor exceeds a certain frequency, photons absorbed by the semiconductor give bound electrons enough energy to jump into the conduction band. The resulting free electrons conduct electricity, thereby lowering resistance. The resistance range and sensitivity of a Photoresistor can substantially differ among dissimilar devices. Moreover, unique Photoresistor may react substantially differently to photons within certain wavelength bands.

A photoelectric device can be either intrinsic or extrinsic. An intrinsic semiconductor has its own charge carriers and is not an efficient semiconductor, for example, silicon. In intrinsic devices, the only available electrons are in the valence band, and hence the photon must have enough

energy to excite the electron across the entire bandgap. Extrinsic devices have impurities, also called dopants, and added whose ground state energy is closer to the conduction band; since the electrons do not have as far to jump, lower energy photons (that is, longer wavelengths and lower frequencies) are sufficient to trigger the device. If a sample of silicon has some of its atoms replaced by phosphorus atoms (impurities), there will be extra electrons available for conduction. This is an example of an extrinsic semiconductor. [6]

The most common type of LDR has a resistance that falls with an increase in the light intensity falling upon the device which is shown in an image below:

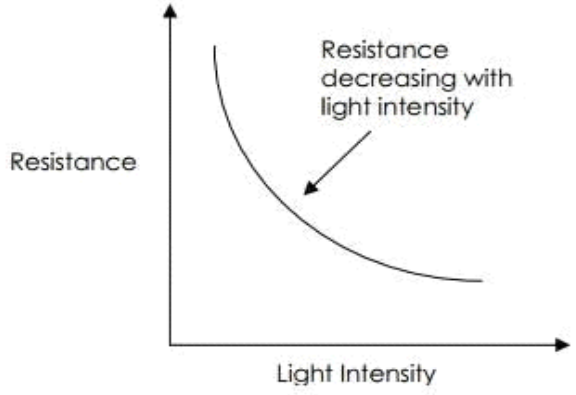

Figure 2.9: Typical LDR resistance vs light intensity graph

The resistance of an LDR may typically have the following resistances:

- Daylight = 5000Ω
- Dark = 20000000Ω

2.19 Applications of LDR

There are many applications for Light Dependent Resistors. They include:

- **Lighting Switch:** The most obvious application for an LDR is to automatically turn on a light at a certain light level. An example of this could be a street light or a garden light.
- **Camera Shutter Control:** LDRs can be used to control the shutter speed on a camera. The LDR would be used to measure the light intensity which then adjusts the camera shutter speed to the appropriate level.

2.20 Technical Specifications of LDR:

- **Resistor Model:** GL5516
- **Size:** 5mm * 2mm
- **Maximum Voltage:** 150 volt DC
- **Maximum Wattage:** 90mW
- **Operating Temperature:** -30 ~ + 70 deg C
- **Spectral Peak:** 540 nm
- **Light Resistance(At 10 Lux):** 5-10 K ohm
- **Dark Resistance:** .5 M ohm
- **Response time:** 20ms(Rise), 30(Down)
- **Resistance Illumination:** 4

2.21 Servo Motor

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity, and acceleration. It consists of a suitable motor coupled to a sensor for

position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servo motors.

Servomotors are not a specific class of motor although the term servo motor is often used to refer to a motor suitable for use in a closed-loop control system.

Servo motors are used in applications such as robotics, CNC machinery or 1automated manufacturing.

2.22 Infrastructure of Servo Motor

Inside of a servo motor, there is a pretty simple setup: a small DC motor, potentiometer, and a control circuit. The motor is attached by gears to the control wheel. As the motor rotates, the potentiometer's resistance changes, so the control circuit can precisely regulate how much movement there is and in which direction.

When the shaft of the motor is at the desired position, the power supplied to the motor is stopped. If not, the motor is turned in the appropriate direction. The desired position is sent via electrical pulses through the signal wire. The motor's speed is proportional to the difference between its actual position and desired position. So if the motor is near the desired position, it will turn slowly, otherwise, it will turn fast. This is called proportional control**.** This means the motor will only run as hard as necessary to accomplish the task at hand, a very efficient little guy. [7]

2.23 Mechanism of Servo Motor

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analog or digital) representing the position commanded for the output shaft.

The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servo motors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servo motor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

More sophisticated servo motors use optical rotary encoders to measure the speed of the output shaft and a variable-speed drive to control the motor speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servomotor to be brought to its commanded position more quickly and more precisely, with less overshooting.

2.24 Types of Servo Motor

There are two types of servo motors: AC servo motor and DC servo motor. AC servo can handle higher current surges and tend to be used in industrial machinery. DC servos are not designed for high current surges and are usually better suited for smaller applications. Generally speaking, DC motors are less expensive than their AC counterparts. These are also servo motors that have been

built specifically for continuous rotation, making it an easy way to get your robot moving. They feature two ball bearings on the output shaft for reduced friction and easy access to the rest-point adjustment potentiometer.

2.25 Servo Library

This library allows an Arduino board to control RC servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite () (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

In our proposed system, we have used **Micro Servo.** This TowerPro SG90 Micro servo with 1.5kg/cm stall torque is a good choice for your RC airplanes. It comes with servo arms and screws for easy installation.

Figure 2.10: SG90 1.5 kg /0.3sec 9g Micro Servo

2.26 Technical Specification of Servo motor:

- **Brand:** TowerPro
- **Model:** SG90
- **Type:** 9g servo *1
- **Dimension:** 23*12.2*29mm.
- **Stall Torque:** 1.5kg/cm.
- **Operating Speed:** 0.3sec/60 degree (4.8V).
- **Operating Voltage:** 4.2-6V.
- **Temperature Range: 0˚C -55˚C.**
- **Dead Band Width:** 10usec.
- **Accessories Include** servo arms and screws.

2.27 Resistors:

In our proposed system, as we have used LDR sensor. LDR sensor needs a 100KΩ resistor.

Figure 2.11: 100KΩ Resistor

2.28 Jumper:

In our proposed system, we have used two types of wires. Male-to-Male jumper and Male-to-Female jumper. Both are shown below:

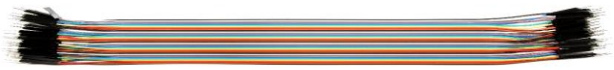

Figure 2.12: Male to Male Jumper

Figure 2.13: Male to Female Jumper

2.29 Schematic diagram of the system

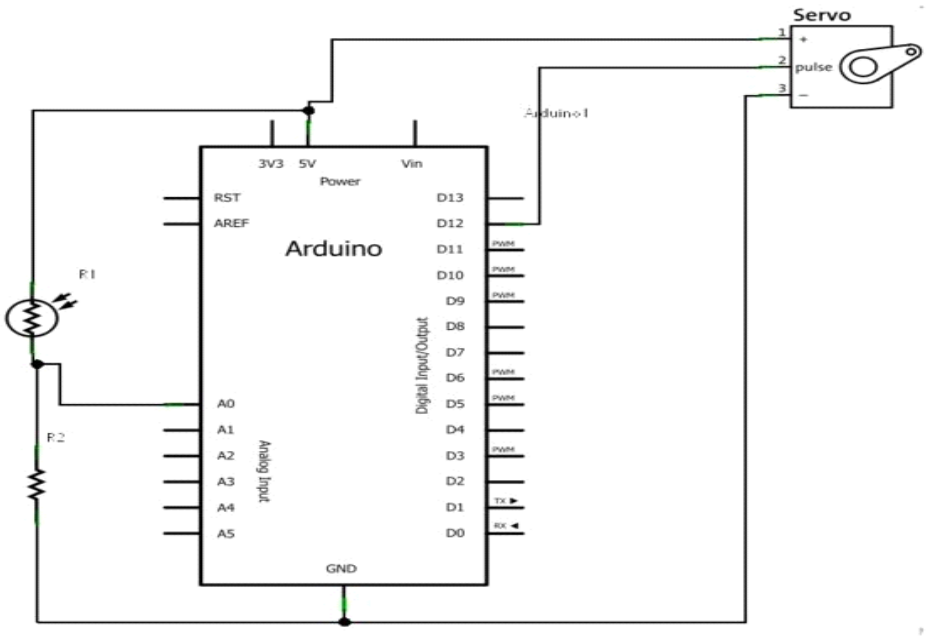

Figure 2.14: Schematic Diagram

Chapter 3 Proposed System Structure

3.1 Design Overview

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development. [10]

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.

In our proposed system, we have designed 2 diagrams named Use Case Diagram and another is Activity Diagram.

- **Use Case Diagram:** A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. So we can say, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

Our proposed systems Use Case Diagram is given below:

Figure 3.1: Use Case Diagram

Here, there are two actors named User and SPARK. SPARK means our software application. In this diagram, there are briefly described the activities of each actor. Any User can be registered by doing registration process and use this application by logging on it. Also, a user can elect space for parking. Then SPARK can show all space to the user and generate PIN. Finally, a user can unlock gate by entering valid PIN to the system which will be validated by SPARK.

- **Activity Diagram:** Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

  Our proposed system's Activity Diagram is given below:

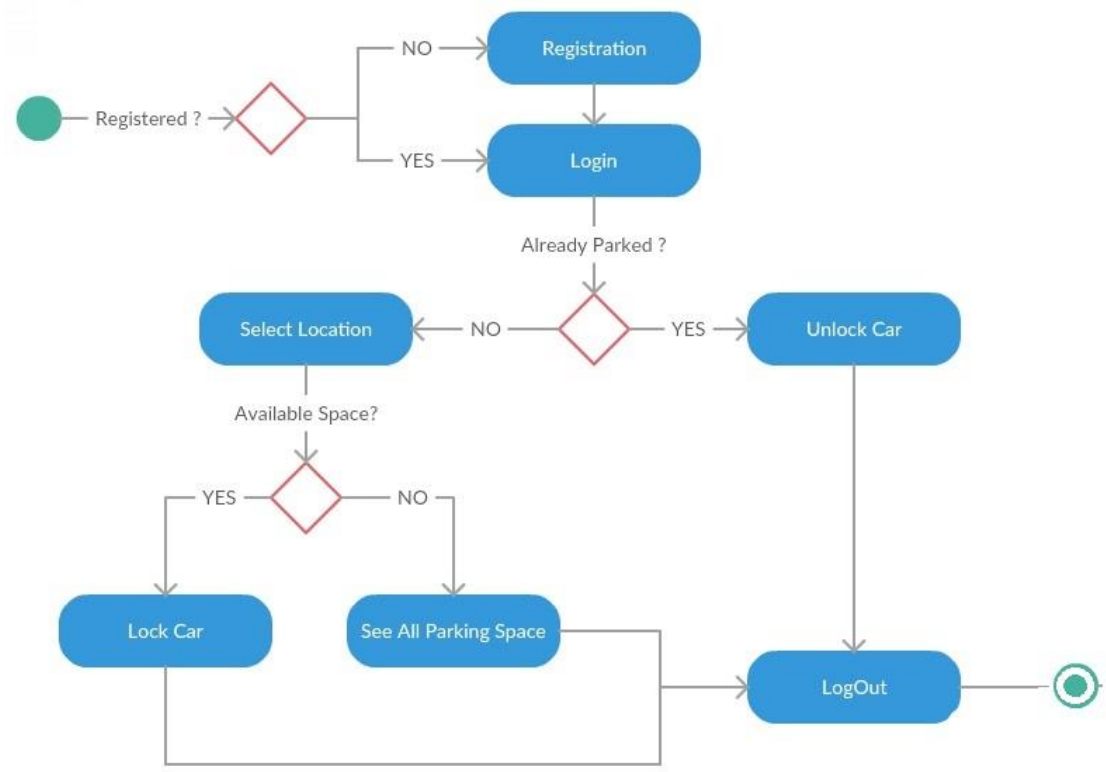

Figure 3.2: Activity Diagram

This diagram shows the users activities to that system. At first, the systems checks whether the user is registered or not, if not registered, after getting registered the system passes the user to login page, on the other hand, if the user is registered, directly go to the login page. After logging to the system, the user can see the space provided for parking by this system with availability or not. After parking

the users' vehicle, h/she can unlock his/her vehicle by giving valid PIN. Finally, h/she can log out from the system.

3.2 Code Structure

The basic structure of the Arduino language is fairly simple and runs in at least two parts. These two parts or functions or block of statements are required. These are:

```
Void Setup()
{
        Statements;
}

Void Loop()
{
        Statements;
}
```

Where setup() is the preparation and loop() is the execution part. Both functions are required for the program to work.

The setup() function should follow the declaration of any variables at the beginning of the program. It is the first function to run the program and it runs only once and is used to set pinMode or initialize serial communication.

The loop() function follows next and includes the code to be executed continuously- reading inputs, triggering outputs etc. This function is the core of all Arduino programs and does the bulk of the work.

3.3 Arduino Initial

#**Include <SPI.h>:** It's a library file for Ethernet Shield. **SPI** -> Serial Peripheral Interface. This library allows you to communicate with SPI devices, with the Arduino as the master device.

Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances. It can also be used for communication between two microcontrollers. With an SPI connection, there is always one master device (usually a microcontroller) which controls the peripheral devices. [8]

Typically there are three lines common to all the devices:

- **MISO** (Master In Slave Out) - The Slave line for sending data to the master.
- **MOSI** (Master Out Slave In) - The Master line for sending data to the peripherals.
- **SCK** (Serial Clock) - The clock pulses which synchronize data transmission generated by the master
- **SS** (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

#Include <Ethernet.h>: This is also a library for Ethernet Shield.The libraries allow an Arduino board to connect to the internet. The board can serve as either a server accepting incoming connections or a client making outgoing ones. The libraries support up to four concurrent connection (incoming or outgoing or a combination). Ethernet library (Ethernet.h) manages the W5100 chip, while Ethernet2 library (Ethernet2.h) manages the W5500 chip; all the functions remain the same.

Arduino communicates with the shield using the SPI bus. This is on digital pins 11, 12, and 13 on the Uno and pins 50, 51, and 52 on the Mega. **On both boards, pin 10 is used as SS.** On the Mega, the hardware SS pin, 53, is not used to select the W5100, but it must be kept as an output or the SPI interface won't work. [9]

#Include <Servo.h>: This library should be included when servo motor used in Arduino. As we have used the servo motor to pull up the gate or pull down the gate, we used the micro servo motor.

This library allows an Arduino board to control RC (hobby) servo motors. The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

Servo micro servo: By this line of code, we have created an object of Servo Library.

int myRand[4]: Created an integer typed array which size is four.

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }: In our Arduino Code, there are some portion which will find my mac address. Then the founded mac address will be inserted into mac[] variable

byte ip[] = { 192, 168, 1, 5 }: Initialized my IP address of my LAN to the Arduino.

EthernetServer server(80): Created an object of Ethernet Server named server and initialized the port address 80.

3.4 Arduino Setup Module

The setup() function is called when the sketch begins. Utilize it to introduce factors, stick modes, begin utilizing libraries, and so on. The setup function will just run once, after each catalyst or reset of the Arduino board.

3.5 Arduino Loop

In the wake of making a setup() function, which in states and sets the underlying esteem, the loop() function does definitely what its name proposes, and loops continuously, enabling your program to change and react. Utilize it to effectively control the Arduino board.

3.6 Functions

- **Server(80):** Connect the Ethernet Client to port address 80.
- **Serial.begin():** Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.
- **Servo.attach():** Attach the servo variable to a specific pin of Arduino. In our proposed system, we have used four servos.
- **Servo.write():** Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).In our proposed system, as we used a standard servo, we wrote the angle of the servo in write(). For example servo.write(90), which means, make the servo in 90 degrees.
- **Serial.available():** Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes).
- **Client.Connected():** Checks whether the client is connected or not. The client is considered connected if the connection has been closed but still there is unread data.
- **Client.available():** Returns the number of bytes available for reading (The amount of data that has been written to the client by the server it is connected to). This function inherits from Stream utility class.

- **Client.read():** Read the next byte received from the server the client is connected to (after the last call to read().This function inherits from Stream utility class
- **String.length():** Returns the length of the string in characters.
- **Serial.println():** Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is.
- **Client.println():** Print data, followed by a carriage return and newline, to the server a client is connected to. Prints numbers as a sequence of digits, each an ASCII character.
- **analogRead():** Reads the value from the specified analog pin. The Arduino board contains a 16 channel on the Mega, 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023.
- **Delay():** Pauses the program for the amount of time (in millisecond's) specified as parameter. E.g.: delay (100), pause the program for 100 milliseconds.
- **Client.stop():** Disconnect the client from server.
- **String.indexOf():** Locates a character or String within another String. By default, searches from the beginning of the String, but can also start from a given index, allowing for the locating of all instances of the character or String.

Chapter 4 System Evaluation

4.1 Login UI

The user has to fill up the login page to access the system.

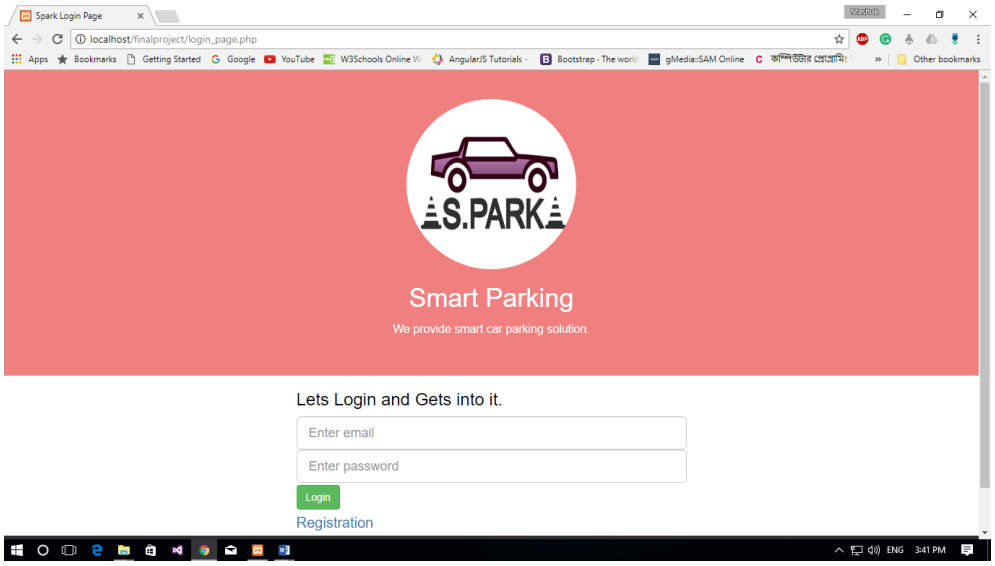

Figure 4.1: Login Page

4.2 User Registration UI

The user needs to be registered through user registration page.

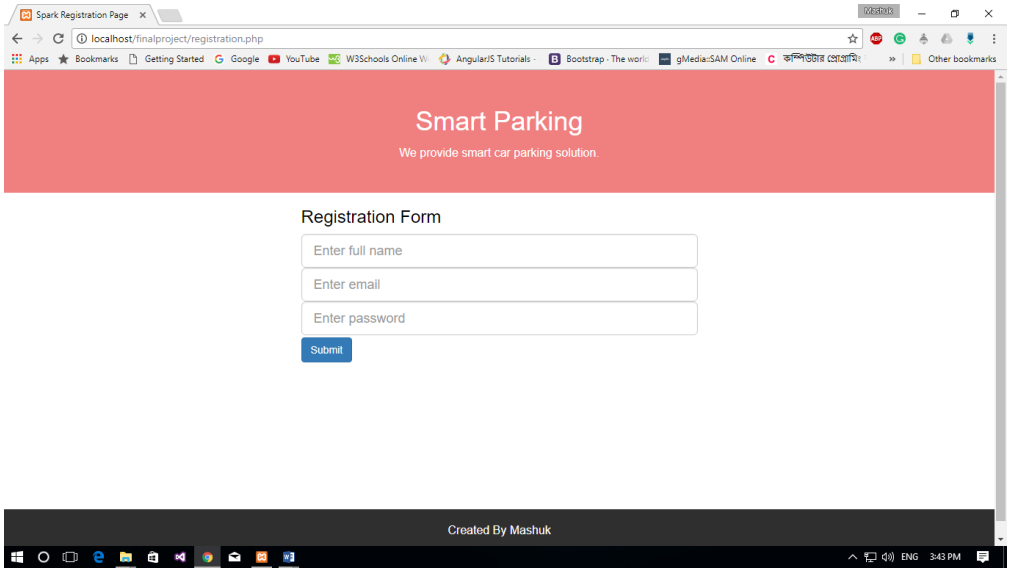

Figure 4.2: Registration Page

4.3 Information Page

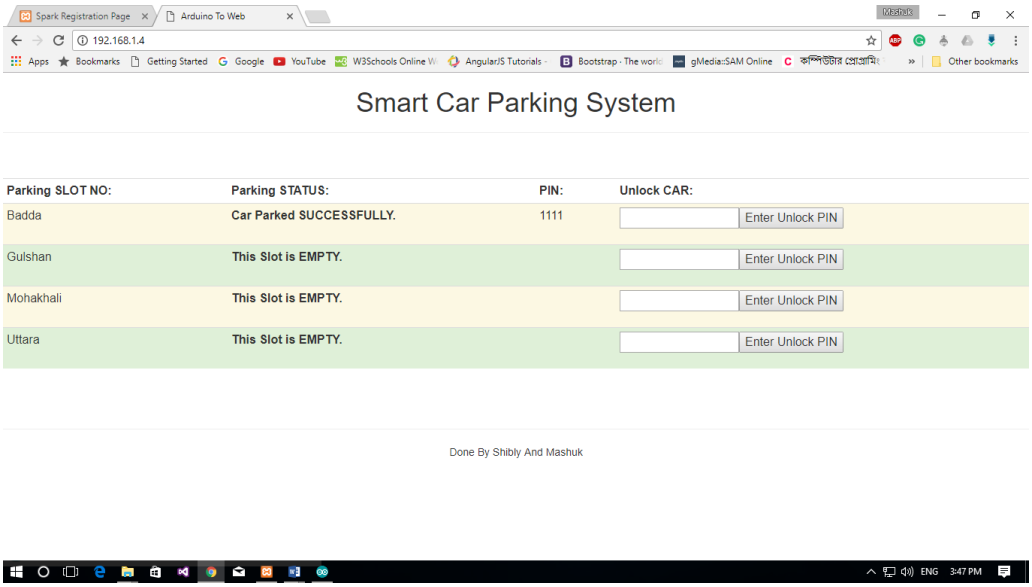

Figure 4.3: Information Page showing "car parked successfully"

After logging in, a user will be redirected to the main page, where a user can see all spaces for parking provided by this system. When any vehicle is parked there, Parking Status will be shown as "Car Parked Successfully" and a Secret PIN number will be generated. This scenario is demonstrated in **Figure 4.3**.

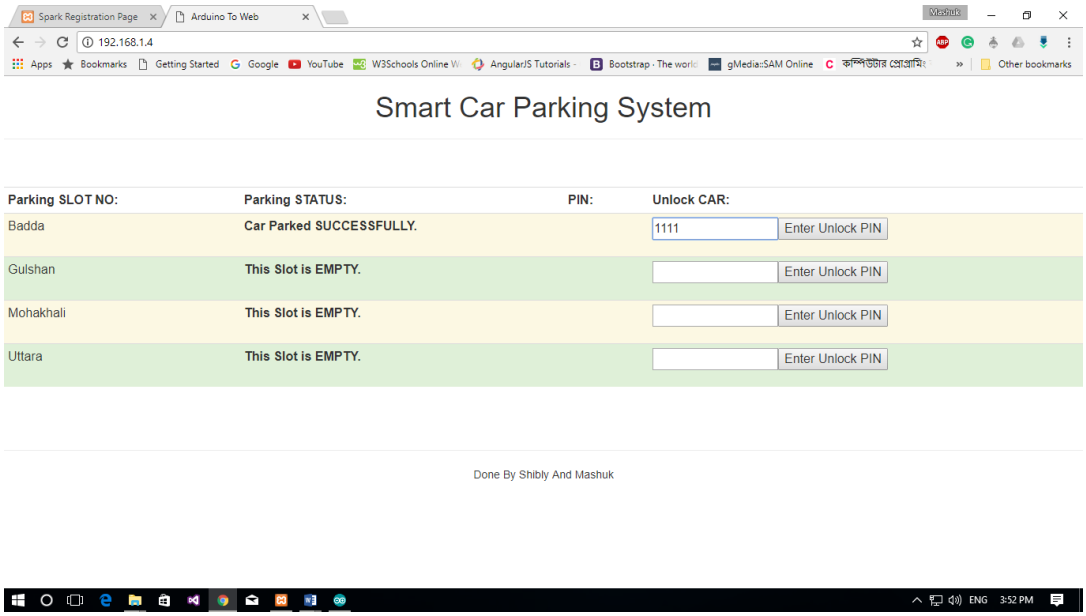

Figure 4.4: Entering PIN number on the Information Page

For Unlocking the vehicle, the user has to enter the PIN in the text field which was generated previously. This is showed on **Figure 4.4**.

If the PIN does not match, the gate will not open, and a “WRONG PIN” message will be shown. Which is given in **Figure 4.5**.

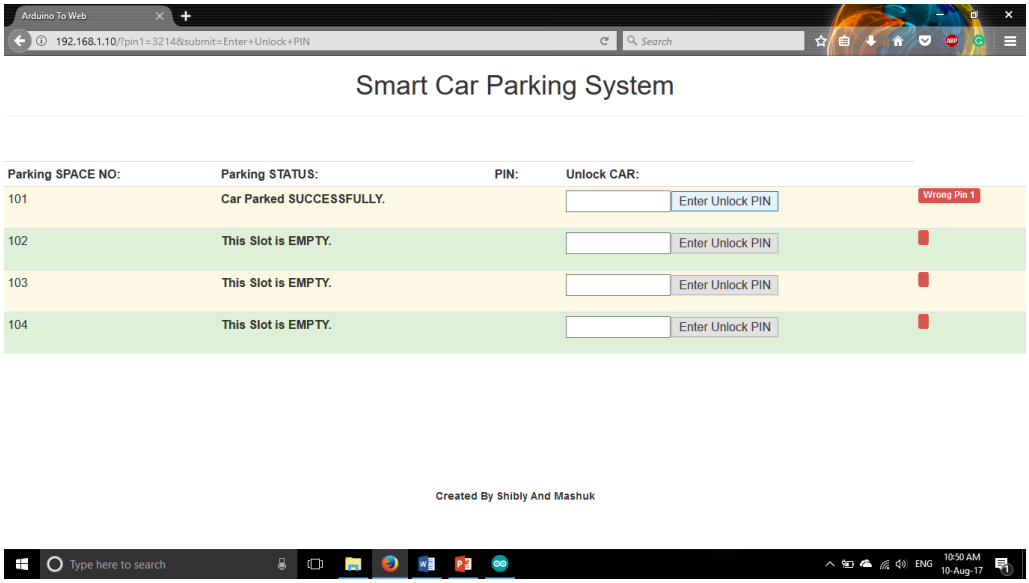

Figure 4.5: Entering wrong PIN number on the Information Page

If the PIN matched, the gate will open and he can get his/her vehicle back. The gate lock and unlock is demonstrated in **Figure 4.6**.

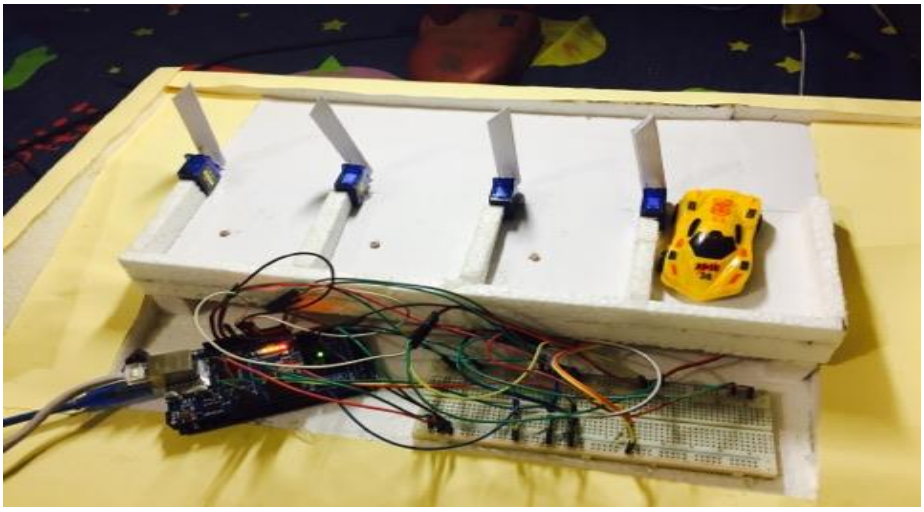

Figure 4.6: Entering PIN number and gate is unlocked

Chapter 5 Conclusion

5.1 Conclusion and Future Work

In future, we will work with real life scenario with more efficiency. Our system will be worked with many places with many sensors. There will be a distributed database to control all the places which will have a domain address. We have implemented partially from a broad aspect. There will be sub-section of control for each parking lot that communicates seamlessly with the main control section.

An important approach that suggests user nearby parking lots if they find no parking space as they expected will be taken. Working with the current location that deals with GPS connectivity will be a necessary step. A GPS module that sends current location of the user helps to bind the surrounding location information.

There should be a business idea that aims to make it profitable. There will be a parking cost per hour which will not exceed a certain limit for users.

We have the intention to make this system more secure by providing strong authentication. There can be messaging service that informs the user where their car is parked and the service charges.

Nevertheless, accessing this service via a web browser is not a good practice. A mobile app would be a better solution as well as a smart way for nowadays. So we have been looking into it.

References

[1]Daily Star, "Traffic Congestion: Tk 20,000cr loss to economy a year", p. 1, 2016.

[2]"Arduino - Introduction", *Arduino.cc*, 2017. Available: https://www.arduino.cc/en/Guide/Introduction. [accessed on 01 August 2017]

[3]"Arduino - ArduinoBoardMega", *Arduino.cc*, 2017. Available: https://www.arduino.cc/en/Main/arduinoBoardMega. [accessed on 01 August 2017]

[4]A. 2560, "Arduino Mega 2560", *Bdspeedytech.com*, 2017. Available: http://bdspeedytech.com/index.php?route=product/product&product_id=1725&search=arduino+mega. [accessed on 01 August 2017]

[5]2017. Available: https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf.

[6] "How an LDR (Light Dependent Resistor) Works | Kitronik", *Kitronik.co.uk*, 2017. Available: https://www.kitronik.co.uk/blog/how-an-ldr-light-dependent-resistor-works/. [accessed on 01 August 2017]

[7]"How Servo Motors Work", *Jameco.com*, 2017.Available: http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html. [accessed on 01 August 2017]

[8]"Arduino - SPI", *Arduino.cc*, 2017. Available: https://www.arduino.cc/en/Reference/SPI. [accessed on 01 August 2017]

[9]"Arduino - Ethernet", *Arduino.cc*, 2017. Available: https://www.arduino.cc/en/Reference/Ethernet. [accessed on 01 August 2017]

[10]"UML Diagram - Everything You Need to Know About UML Diagrams", *Smartdraw.com*, 2017. Available: https://www.smartdraw.com/uml-diagram/. [accessed on 01 August 2017]

Appendix A. Arduino Code

Code for LDR sensor checking and Servo motor movement:

```
int LDR1 = analogRead(A0);


        if(LDR1<150)
        {
          result1="Car Parked SUCCESSFULLY.";
         delay(100);
           microservo1.write(90);


           if(check1==1)
           {
             Row1=1;
              check1++;
             }
         }


         else
         {
           check1=1;
           result1="This Slot is EMPTY.";
           delay(100);
           microservo1.write(0);
```

```
    }


String result2;
int LDR2 = analogRead(A1);


if(LDR2<200)
{
  result2="Car Parked SUCCESSFULLY.";
   microservo2.write(90);
   if(check2==1)
    {


      Row2=1;
       check2++;
      }
  }
  else
  {
    check2=1;
    result2="This Slot is EMPTY.";
     microservo2.write(0);
    }
```

```
 String result3;
int LDR3 = analogRead(A2);

if(LDR3<100)
{
  result3="Car Parked SUCCESSFULLY.";
  microservo3.write(90);
   if(check3==1)
    {

      Row3=1;
       check3++;
      }
  }
  else
  {
     check3=1;
    result3="This Slot is EMPTY.";
    microservo3.write(0);
    }

String result4;
int LDR4 = analogRead(A3);

if(LDR4<150)
{
```

```
result4="Car Parked SUCCESSFULLY.";
microservo4.write(90);
 if(check4==1)
  {

    Row4=1;
     check4++;
    }
}
else
{
  check4=1;
  result4="This Slot is EMPTY.";
  microservo4.write(0);
  }
```

Code for Pin generator:

```
client.println("<td>");
        if(Row1==1)
        {
         client.print(myRand[0]);
         Row1++;
        }
                client.println("</td>");
```