



EAST WEST UNIVERSITY

Automated Toll Collection System

Submitted By

Md. Jahidul Islam
ID: 2012-2-60-024

Dewan Mofasser Hossain
ID: 2012-3-60-001

Supervised By

Dr. Anisur Rahman
Assistant Professor
Department of Computer Science & Engineering
East West University

The project has been submitted to the Department of the Computer Science & Engineering at East West University in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering.

August, 2017

Declarations

The project has been submitted to the Department of the Computer Science & Engineering, East West University in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering. We hereby, declare that this project has not been submitted elsewhere for the requirement of any degree or diploma or any other purposes.

Signature of the students

.....
(Md. Jahidul Islam)
(ID: 2012-2-60-024)

.....
(Dewan Mofasser Hossain)
(ID: 2012-3-60-001)

Letter of Acceptance

This project is entitled “Automated Toll Collection System” submitted by: Md. Jahidul Islam ID: 2012-2-60-024 and Dewan Mofasser Hossain ID: 2012-3-60-001 to the department of Computer Science & Engineering, East West University, Dhaka, Bangladesh is accepted as satisfactory for partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering on August 2017.

Supervisor

.....
Dr. Anisur Rahman
Assistant Professor
Department of Computer Science & Engineering
East West University, Dhaka, Bangladesh.

Chairperson

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

Abstract

Due to the expansion of the vehicle transportation system, Jam on toll collection booth is common nowadays & become a serious problem in recent years. Because of increase lots of vehicle and toll are collected by human, lots of time waste is occurring here, so it is essential to design & implement a modern system, which can monitor & collect toll from these vehicles automatically. A system that is able to sense the vehicle and collect toll from the vehicle automatically. So, human toll collector is not needed here and vehicle don't need to waste time by waiting on toll booth because it will collect toll faster than human toll collector.

In our project, we have showed it by interfacing stepper motor, sonar sensor and Arduino Uno microcontroller. When a vehicle arrives, if the car owner has registered RFID card and sufficient balance on his RFID card, he just need to punch it on RFID sensor, the other work will be done by our system automatically like deduct actual toll amount from RFID balance. Also, there will be two sonar sensor to sense the arriving and leaving of the car. So, it will bring automation on barrier which have made by stepper motor. When a car arrives to the toll booth, the first sonar sensor will sense that, car have arrived and sensor will wait for the car to punch RFID card on RFID sensor and start counting the time and if the car owner don't punch RFID card on RFID sensor because of haven't any registered RFID card or haven't sufficient amount of balance on RFID card or any other reason, system have Buzzer that will make a loud sound of alarm to make the car owner alert to punch RFID card or go left for pay manually and a message will show on LCD display named Please Go Left and left sign for the better understand of the car owner, Then it have to go for manual pay and pay the amount of toll with some amount of fine. But, if the car owner has registered RFID card and sufficient amount of balance and punch on RFID sensor, the barrier will rise up by 90 degrees and wait for the car to go and the second sonar sensor will sense the leaving of the car and when it passes away the motor barrier will go down by 90 degrees.

Actually, the rise and down of stepper motor barrier will also be automated. So, any human operator isn't needed here to operate the stepper motor barrier as well as all the system will be automated and reduce time waste on toll booth.

Acknowledgement

First of all, we would like to thank almighty Allah for giving us strength, patience and knowledge to complete this project work.

Our most heartfelt gratitude goes to our beloved parents for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to end.

We would like to express our sincere gratitude to our supervisor Dr. Anisur Rahman for the continuous support, for his patience, motivation, and immense knowledge. His guidance helped us in all the time of our work. We could not have imagined having a better supervisor and mentor for this project.

We would like to express our sincere gratefulness to the faculty members of the Department of Computer Science and Engineering (CSE), East West University, Bangladesh for their friendly attitude and enthusiastic support that has given us and helped us with their feedback.

In addition, we would like to thank our friends and seniors for their motivation, stimulation, feedback, cooperation and of course friendship.

TABLE OF CONTENTS

Title	Page no
Declaration	i
Letter of Acceptance	ii
Abstract	iii
Acknowledgement	iv
Table of Contents	v-viii
Chapter 01	
Introduction	01-04
1 Introduction	01
1.1 Motivation	02
1.2 Objective	03
1.3 Contribution	04
1.4 Outline	04
Chapter 02	
Background	05-08
2.1 What is Arduino?	05
2.2 Why Arduino?	06
2.3 What can we do with Arduino?	07
2.4 What We Need for a Working System?	07
2.5 Installing the Software	07
2.6 Arduino Programming Language	08
2.7 Common Coding Errors	08
Chapter 03	
Description of the Hardware System	9-34
3.1 Arduino Uno	9
3.2 RFID (MRC522) sensor and card	16
3.3 Servo Motor	22
3.4 LCD Display 16x2	25
3.5 IIC/I2C Serial Interface Module	27
3.6 Sonar Sensor	28
3.7 Buzzer	30

Chapter 04		
Description of the Software System		35-44
4.1	Introduction	35
4.2	Structure	36
4.3	Variable Declaration	37
4.4	Arithmetic Operator	37
4.5	Compound Operator	38
4.6	Boolean Operator	39
4.7	pinMode(pin,mode)	40
4.8	digitalWrite(pin,value)	41
4.9	delay(ms)	42
4.10	Serial.begin(speed)	42
4.11	Serial.println(val)	43
Chapter 05		
Conclusion		45-46
5.1	Future Work	46
References		47-48
Appendix		
Code		49-54

List of Figures:

Fig 1.1	Toll collection booth at kanchpur meghna bridges and jam on toll collection booth	03
Fig 3.1	Arduino Uno	11
Fig 3.2	RFID-RC522	17
Fig 3.3	Simplified MFRC522 Block diagram	20
Fig 3.4	RFID tag 1	21
Fig 3.5	RFID tag 2	21
Fig 3.6	Servo motor	22
Fig 3.7	Servo motor Dimensions and specifications	23
Fig 3.8	Pinout Diagram	23
Fig 3.9	16x2 LCD Display	25
Fig 3.10	Pin Diagram	26
Fig 3.11	Schematic diagram of Arduino Uno & LCD Display	27
Fig 3.12	IIC/I2C Serial Interface Module	28
Fig 3.13	Sonar Sensor	29
Fig 3.14	Buzzer	30
Fig 3.15	Output 1	31
Fig 3.16	Output 2	31
Fig 3.17	Output 3	31
Fig 3.18	Output 4	31
Fig 3.19	Hardware Connections	32
Fig 3.20	Schematic Diagram	33
Fig 3.21	Prototype of the toll system	34
Fig 4.1	Output of serial monitor 1	44
Fig 4.2	Output of serial monitor 2	44

List of Tables:

Table 1.1	Yearly Bi-directional Flow Variation on Meghna-Gomoti Bridge, 2006-2014	02
-----------	---	----

Chapter 1

Introduction

Since Bangladesh is developing day by day, its road and transport system is also developing and as a result, the number of vehicle in Bangladesh is increasing rapidly. The bearing of human and product through vehicle is also raising because people are now using vehicles for many purposes like travels, business etc. as affordable, comfortable and use road for bearing business product are also very popular for its low cost. As the number of vehicles in Bangladesh is raising day by day, the pressure on toll collection booth is also raising. It is a common matter of huge jam on toll collection booth nowadays. Because, toll is collected by manually and it waste some time, as a result our valuable time are lost and it hampers the productivity of our country. So, we have developed a system where toll collection will be automated and the car just have to have a registered RFID card and sufficient amount on his balance. When a car arrives the first sonar sensor will notice it and if it has no RFID, the buzzer will give alarm and LCD will show a message to go left for manual pay. But, if it has valid RFID card and scan on RFID sensor, it will deduct toll amount and allow this car to go by open barrier created with stepper motor and second sonar sensor will notice the left of that car and close the barrier. To achieve this purpose, the system is controlled by a micro controller which is Arduino Uno R3. Managing toll booths is a very complicated task but using our system that automates the entire toll collection process of toll booth, it will reduce the consumption of time.

1.1 Motivation

The uses of vehicles in Bangladesh are increasing day by day because of the affordable cost on transport; so, people are more frequently uses vehicle for travel and business purpose. But, the management of this huge number of car on toll booth is very challenging for human toll collector. He has to stop the car, collect toll and let the car to go which is very much time consuming and create jam if the number of car in toll booth is huge. What if a system can provide toll collection in a fast way where any human toll collector don't needed.

Year	Yearly Volume
2006	3,210,410
2007	4,313,837
2008	4,732,949
2009	5,632,798
2010	5,915,501
2011	6,045,819
2012	6,239,174
2013	6,464,564
2014	7,018,983

Table 1.1: Yearly Bi-directional Flow Variation on Meghna-Gomoti Bridge, 2006-2014 [19]

For sample, if we see the above table, we can see the yearly Bi-directional Flow Variation of year 2006-2014 on Meghna-Gomoti Bridge, one of the major Bridge situated in

Dhaka Chittagong Highway, we can see that the load of vehicles is raising year by year, as a result load on toll collection booth also be raised in recent years and if we still remain on human-based toll collection system, traffic problem on toll collection booth will not reduce.

So, we can introduce automated toll collection system where don't need any human toll collector. we can use an automated system that can collect toll by RFID card which will be previously recharged and the barrier also be automated. There have RFID sensor, which will sense the RFID card, deduct toll and let the car to go. The rise of vehicle is concern for us and we can introduce this system to reduce this problem.



Figure 1.1: Toll collection booth at kanchpur meghna bridges [18] and jam on toll collection booth. [20]

1.2 Objectives

The main objective of our project is to bring automation on toll collection both. It will collect information from RFID card user and match with stored data. Then if match information it will deduct toll money from that user and allow the car to go away. In Bangladesh, this system hasn't introduced yet. If it can be implemented it will save time and keep the road less busy.

1.3 Contribution

We have used Arduino Uno R3 micro controller in this project. This micro controller is interfacing with computer and its instruction is written on Arduino IDE using C programming language. In our work we showed that how toll collection both can be automated and take toll from car using RFID sensor without human to reduce the jam. In our system, when a car come the RFID sensor mark the car and when it punches RFID card, toll money are deduct and stepper motor rise up the barrier and another sonar sensor sense when it go away and down the barrier.

1.4 Outline

Chapter 1: We have discussed about our motivation to do this work, which is our objective and the contribution we have made.

Chapter 2: Brief discussion of the backgrounds where we discuss what types of software, microcontroller, programming language we implement in our project.

Chapter 3: Briefly describes the hardware parts of our system that we have used.

Chapter 4: Describes the software parts of our system. The code we wrote to build this system have described part by part here.

Chapter 5: Concludes the work and future plan, what can be improve in future are written here and the summarization of our work.

Chapter 2

Background

2.1 What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. We can tell our board what to do by sending a set of instructions to the microcontroller on the board. To do so we use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.[1]

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their

particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

2.2 Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

- It is an open-source project, software/hardware is extremely accessible and very flexible to be customized and extended
- It is flexible, offers a variety of digital and analog inputs, *SPI* and serial interface and digital and *PWM* outputs
- It is easy to use, connects to computer via USB and communicates using standard serial protocol, runs in standalone mode and as interface connected to PC/Macintosh computers.

2.3 What can we do with Arduino?

Arduino is a great tool for developing **interactive objects**, taking inputs from a variety of switches or sensors and controlling a variety of lights, motors and other outputs. Arduino projects can be stand-alone or they can be connected to a computer using USB. The Arduino will be seen by the computer as a standard serial interface. There are serial communication APIs on most programming languages so interfacing Arduino with a software program running on the computer should be pretty straightforward.

2.4 What We Need for a Working System?

1. Arduino Uno board
2. USB programming cable (A to B)
3. 9V battery or external power supply (for stand-alone operation)
4. Solderless breadboard for external circuits, and 22 g solid wire for connections
5. Host PC running the Arduino development environment. Versions exist for Windows, Mac and Linux.

2.5 Installing the Software

Follow the instructions on the Getting Started section of the Arduino web site,

<http://arduino.cc/en/Guide/HomePage>. Go all the way through the steps to where we see the pin 13 LED blinking. This is the indication that we have all software and drivers successfully installed and can start exploring with our own programs.

2.6 Arduino Programming Language

The Arduino runs a simplified version of the C programming language, with some extensions for access the hardware. In this guide, we will cover the subset of the programming language that is most useful to the novice Arduino designer. For more information on the Arduino language, see the Language Reference section of the Arduino web site,

<http://arduino.cc/en/Reference/HomePage>.

All Arduino instructions are one line. The board can hold a program hundreds of lines long and has space for about 1,000 two-byte variables. The Arduino executes programs at about 300,000 source code lines per sec.

2.7 Common Coding Errors

By writing code we face some common coding errors as like forgetting the semi-colon at the end of a statement, misspelling a command, forgetting opening or closing brackets. At that moment we face coding errors in our program so we must have sincere about this issues while coding.

Chapter 3

Description of Hardware System

3.1 Arduino Uno

The **Arduino Uno** is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. [2]

The origin of the Arduino project started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy. At that time, the students used a BASIC Stamp microcontroller at a cost of \$100, a considerable expense for many students. In 2003 Hernando Barragán created the development platform *Wiring* as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas, who are known for work on the Processing language. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.[3]

The first Arduino was introduced in 2005, based on 8-bit Atmel AVR, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermo-stats and motion detectors.

An Arduino board consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed *shields*. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel mega AVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants), although some designs such as the Lily Pad run at 8 MHz and dispense with the on-board voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external chip programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, opt boot boot loader is the default boot loader installed on Arduino UNO.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip

with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used.

Arduino and Arduino-compatible boards use printed circuit expansion boards called *shields*, which plug into the normally supplied Arduino pin headers. Shields can provide motor controls for 3D printing and other applications, Global Positioning System (GPS), Ethernet, liquid crystal display (LCD), or bread boarding (prototyping). Several shields can also be made do it yourself (DIY).

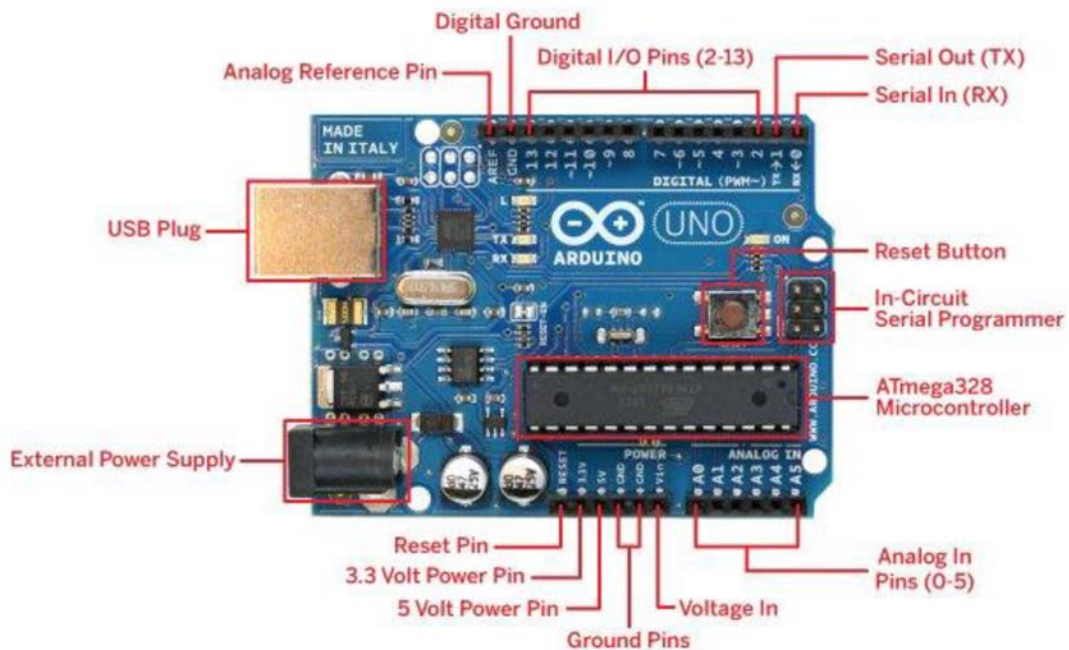


Figure 3.1: Arduino Uno

Technical Specification

- Microcontroller ATmega328
- Operating Voltage 5V
- Input Voltage (recommended) 7-12V
- Input Voltage (limits) 6-20V
- Digital I/O Pins 14 (of which 6 provide PWM output)
- Analog Input Pins 6
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32 KB of which 0.5 KB used by bootloader
- SRAM 2 KB
- EEPROM 1 KB
- Clock Speed 16 MHz

The Arduino Uno 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 centre-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.

The power pins are as follows

- **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 other 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 regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the boot loader; It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM Library).

Input & Output

Each of the 14 digital pins on the Uno 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 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the `analogWrite()` function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **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. The Uno has 6 analog inputs, each of which provide 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 the `analogReference()` function. Additionally, some pins have specialized functionality:
- **I2C: 4 (SDA) and 5 (SCL).** Support I2C (TWI) communication using the Wire library. There are a couple of other pins on the board:
- **AREF.** 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.

Arduino LEDs

Likewise, the Arduino has **four** LEDs: **L**, **RX**, **TX**, and **ON**. **On** the UNO, three are in the middle and one is to the right.

- **ON** LED - this LED will shine green whenever the Arduino is powered. Always check this LED if Arduino is not acting right, if its flickering or off then check the power supply.
- **RX** and **TX** LEDs - these are like the 'send' and 'receive' LEDs on our cable modem. They blink whenever information is sent from or 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 we 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 turn it on or off before start writing code.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required. 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 USB-to serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

The Arduino IDE (integrated development environment) supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled.

3.2 RFID (MRC522) sensor and card

A radio frequency identification reader (RFID reader) is a device used to gather information from an RFID tag, which is used to track individual objects. Radio waves are used to transfer data from the tag to a reader. RFID is a technology similar in theory to bar codes. However, the RFID tag does not have to be scanned directly, nor does it require line-of-sight to a reader. The RFID tag it must be within the range of an RFID reader, which ranges from 3 to 300 feet, in order to be read. RFID technology allows several items to be quickly scanned and enables fast identification of a particular product, even when it is surrounded by several other items. [4,5]

In 1945, Léon Theremin invented an espionage tool for the Soviet Union which retransmitted incident radio waves with the audio information. Sound waves vibrated a diaphragm which slightly altered the shape of the resonator, which modulated the reflected

radio frequency. Even though this device was a covert listening device, not an identification tag, it is considered to be a predecessor of RFID, because it was likewise passive, being energized and activated by waves from an outside source.[6]

MF RC522 is used in highly integrated 13.56MHz contactless communication card chip to read and write, of NXP for “three” and the application launched a low voltage, low cost, small size, non-contact card chips to read and write, intelligent instruments and portable handheld devices developed better. The MF RC522 use of advanced modulation and demodulation concept completely integrated in the 13.56MHz all kinds of passive contactless communication methods and protocols. 14443A compatible transponder signal. The digital part handles the ISO14443A frames and error detection. In addition, support Quick CRYPTO1 encryption algorithm, the term verification MIFARE series. MFRC522 support MIFARE series of high-speed non-contact communication, two-way data transfer rates up to 424kbit / s.As 13.56MHz highly integrated card reader series chip new family, the MF RC522 MF RC500 MF RC530 there are many similarities, but also have many of the characteristics and differences. Communication between it and the host SPI mode, helps to reduce the connection, reduce PCB board volume and reduce costs.

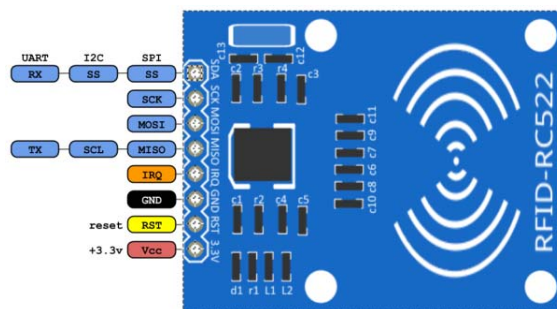


Figure 3.2: RFID-RC522

Features:

- Highly integrated analog circuitry to demodulate and decode responses
- Buffered output drivers to connect an antenna with minimum number of external components
- Supports ISO/IEC 14443A / MIFARE®
- Typical operating distance in Reader/Writer mode for communication to a
- ISO/IEC 14443A / MIFARE® up to 50 mm depending on the antenna size and tuning
- Supports MIFARE® Classic encryption in Reader/Writer mode
- Supports ISO/IEC 14443A higher transfer speed communication up to 848 kbit/s
- Support of the MFIN / MFOUT
- Additional power supply to directly supply the smart card IC connected via MFIN / MFOUT
- Supported host interfaces
- SPI interface up to 10 Mbit/s
- I2C interface up to 400 kbit/s in Fast mode, up to 3400 kbit/s in High-speed mode
- serial UART in different transfer speeds up to 1228.8 kbit/s, framing according to the RS232 interface with voltage levels according pad voltage supply
- Comfortable 64 byte send and receive FIFO-buffer
- Flexible interrupt modes
- Hard reset with low power function
- Power-down mode per software

- Programmable timer
- Internal oscillator to connect 27.12 MHz quartz
- 2.5 - 3.3 V power supply
- CRC Co-processor
- Free programmable I/O pins
- Internal self-test

Specifications:

- Module Name: Rfid-RC522
- Working current : 13—26mA/ DC 3.3V
- Standby current : 10-13mA/DC 3.3V
- sleeping current : <80uA
- peak current : <30mA
- Working frequency : 13.56MHz
- Card reading distance : 0~60mm (mifare1 card)
- Protocol : SPI
- data communication speed : Maximum 10Mbit/s
- Card types supported : mifare1 S50、mifare1 S70、mifare UltraLight、mifarePro、mifareDesfire
- Dimension : 40mm×60mm
- Environment
- Working temperature : -20—80 degree

- Storage temperature : -40—85 degree
- Humidity : relevant humidity 5%—95%
- Max SPI speed: 10Mbit/s

Block Diagram

The Analog interface handles the modulation and demodulation of the analog signals. The contactless UART handles the protocol requirements for the communication schemes in co-operation with the host. The comfortable FIFO buffer allows a fast and convenient data transfer from the host to the contactless UART and vice versa. Various host interfaces are implemented to fulfil different customer requirements.

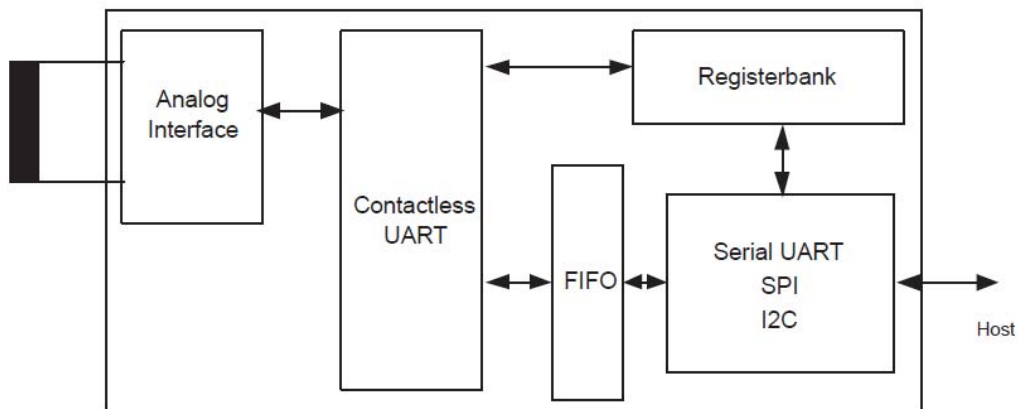


Fig 3.3: Simplified MFRC522 Block diagram.

RFID Card

RFID tagging is an ID system that uses small radio frequency identification devices for identification and tracking purposes. An RFID tagging system includes the tag itself, a read/write device, and a host system application for data collection, processing, and transmission. An RFID tag (sometimes called an RFID transponder) consists of a chip , some memory and an antenna .



Figure 3.4: RFID Tag 1

Every system has to start with selecting an **RFID tag** that will perform well given your reading requirements, size constraints, and application type. A tag consists of two parts: the chipset and antenna. Normally, the larger the antenna the better range you will get. There are a variety of chipsets and antenna designs which are designed to perform in all types of applications.

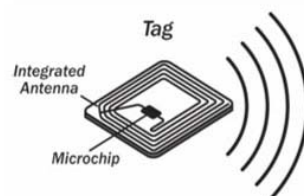


Fig 3.5: RFID tag 2

3.3 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 servomotors. [7,8]

Servomotors are not a specific class of motor although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system. Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.



Fig 3.6: Servo motor.

Servos are controlled by sending an electrical pulse of variable width, or **pulse width modulation** (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movements. The motor's neutral position is defined as the position where the servo has the same amount of potential rotation in the both the clockwise or counter-clockwise direction.

The PWM sent to the **motor** determines position of the shaft, and based on the duration of the pulse sent via the control wire; the **rotor** will turn to the desired position. The servo motor

expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns. For example, a 1.5ms pulse will make the motor turn to the 90° position. Shorter than 1.5ms moves it in the counter clockwise direction toward the 0° position, and any longer than 1.5ms will turn the servo in a clockwise direction toward the 180° position.

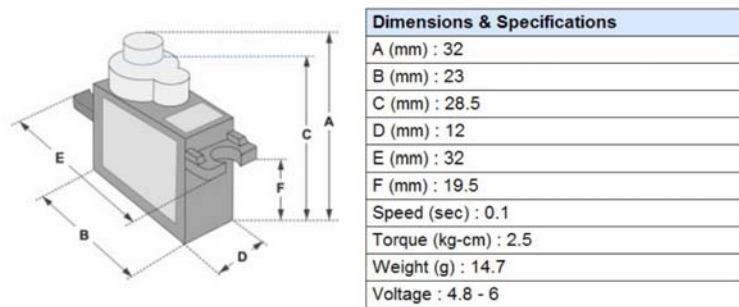


Fig 3.7: Servo motor Dimensions and specifications

Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

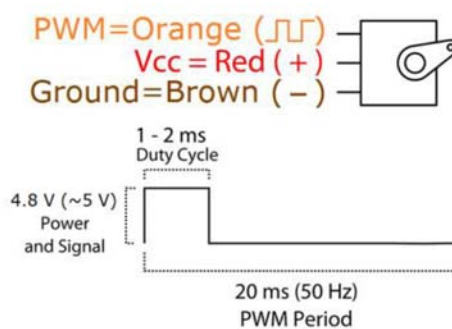


Fig 3.8: Pinout Diagram

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

Types of Servo Motors

There are two types of servo motors - AC and DC. 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**.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf·cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 μ s
- Temperature range: 0 °C – 55 °C

Servo Motor Applications

Servos are used in radio-controlled airplanes to position control surfaces like elevators, rudders, walking a robot, or operating grippers. Servo motors are small, have built-in control circuitry and have good power for their size.

3.4 LCD Display 16x2

A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and 7-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made up of a large number of small pixels, while other displays have larger elements.[8]

This example sketch prints "Hello World!" to the LCD and shows the time in seconds since the Arduino was reset:



Figure 3.9: 16x2 LCD Display

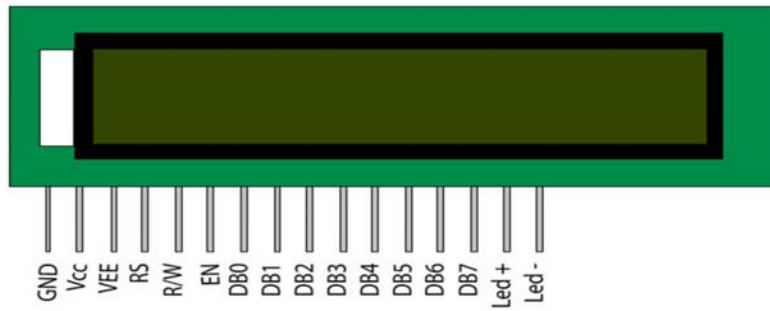


Figure 3.10: Pin Diagram

Pin No	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5V	Supply Voltage for logic
3	VO	(Variable)	Operating voltage for LCD
4	RS	H/L	H: DATA, L: Instruction code
5	R/W	H/L	H: Read(MPU?Module) L: Write(MPU?Module)
6	E	H,H->L	Chip enable signal
7	DB0	H/L	Data bus line
8	DB1	H/L	Data bus line
9	DB2	H/L	Data bus line
10	DB3	H/L	Data bus line
11	DB4	H/L	Data bus line
12	DB5	H/L	Data bus line
13	DB6	H/L	Data bus line
14	DB7	H/L	Data bus line
15	A	5V	LED +
16	K	0V	LED-

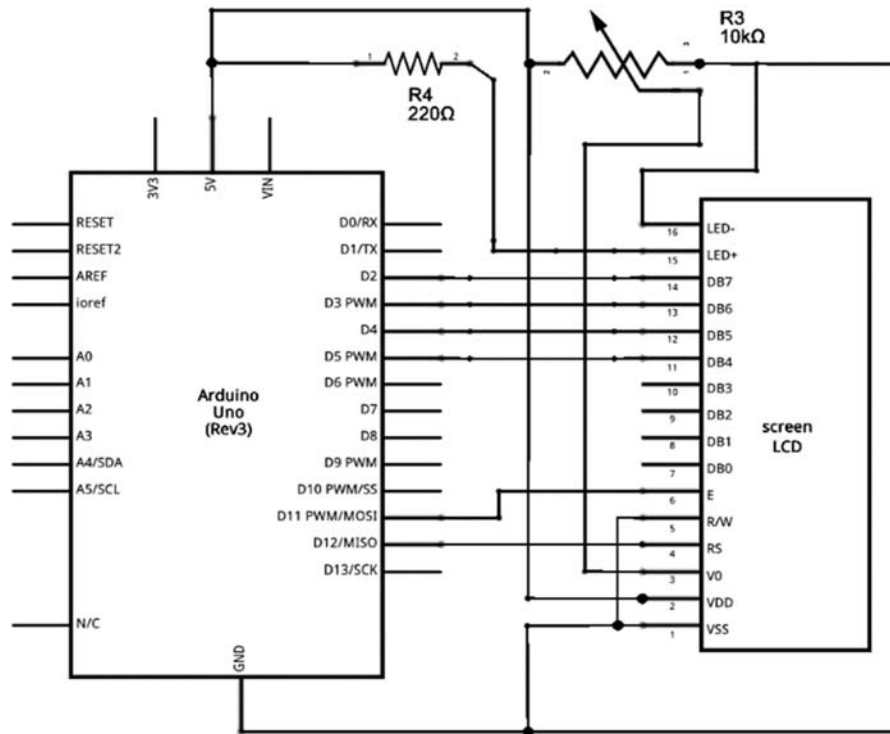


Figure 3.11: Schematic diagram of Arduino Uno & LCD Display

3.5 IIC/I2C Serial Interface Module

This is a RoHS compliant I2C Serial LCD Daughter board that can be connected to a standard HD44780 compatible 16x2 or 20x4 Character Display Module that supports 4-bit mode. All Character Modules sold on our site support 4-bit mode, and nearly all commercially available 16x2 and 20x4 line character modules support it too. [9]

This board has a PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display. There are many examples on internet for using this board with Arduino. Do a search for "Arduino LCD PCF8574". The I2C address is 0x3F by default, but this can be changed via 3 solder jumpers provided on the board.

This allows up to 3 LCD displays to be controlled via a single I2C bus (giving each one its own address).

- 5V power supply
- Serial I2C control of LCD display using PCF8574
- Backlight can be enabled or disabled via a jumper on the board
- Contrast control via a potentiometer
- Can have 8 modules on a single I2C bus (change address via solder jumpers) address, allowing
- Size : 41.6mm x 19.2mm



Figure 3.12: IIC/I2C Serial Interface Module

These serial interface modules simplify connecting an Arduino to a 16x2 Liquid Crystal display using only 4 wires.

3.6 Sonar Sensor

This is the HC-SR04 ultrasonic ranging sensor. This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver and a control circuit.

[11,12]

There are only four pins that you need to worry about on the HC-SR04: VCC (Power), Trig (Trigger), Echo (Receive), and GND (Ground). Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- Using IO trigger for at least 10us high level signal,
- The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- IF the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time×velocity of sound (340M/S) / 2

Pin configuration of sonar sensor:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground



Figure 3.13: Sonar sensor

3.7 Buzzer

A buzzer or beeper is an audio signaling device which maybe mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.[13]

Buzzer is an integrated structure of electronic transducers, DC power supply, widely used in computers, printers, copiers, alarms, electronic toys, automotive electronic equipment, telephones, timers and other electronic products for sound devices. Active buzzer 5V Rated power can be directly connected to a continuous sound, this section dedicated sensor expansion module and the board in combination, can complete a simple circuit design, to "plug and play."

Pin configuration of buzzer:

1. VCC
2. Input
3. Ground



Figure 3.14: Buzzer

LCD 16x2 Output:



Figure 3.15: Output 1



Figure 3.16: Output 2



Figure 3.17: Output 3



Figure 3.18: Output 4

Hardware Connections:

To create this hardware connection we use Fritzing software and by the way we also create its schematic diagram using the same software in the next figure. We add all the sensor's that we use in our project . We use one arduino, one RFID RC 522, one servo motor, two ultrasonic for different purpose of use, one buzzer , one lcd 16x2 display, one breadboard and LED's . We all connect them by wire and designed our hardware connection.

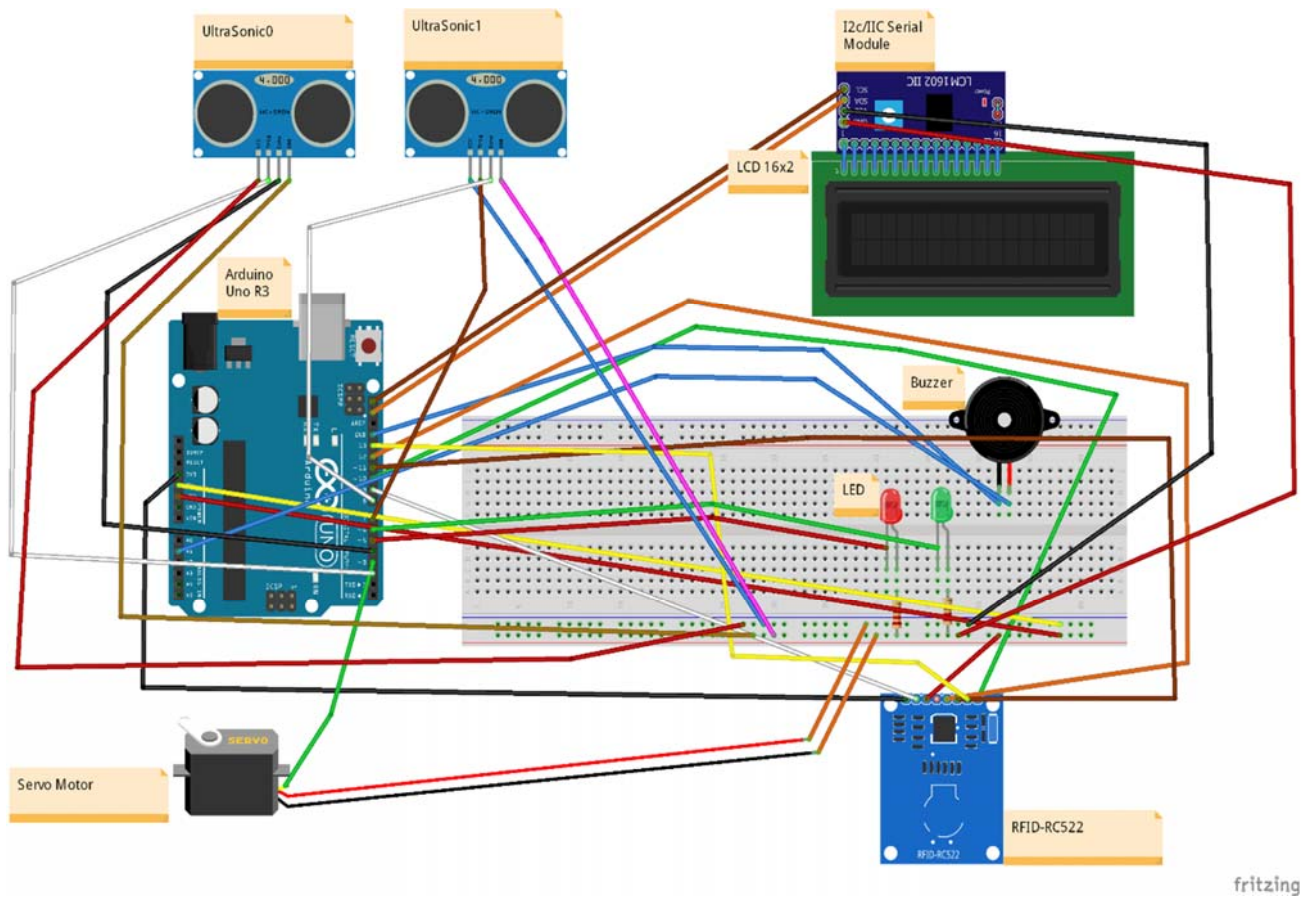


Figure 3.19: Hardware Connections.

Schematic Diagram :

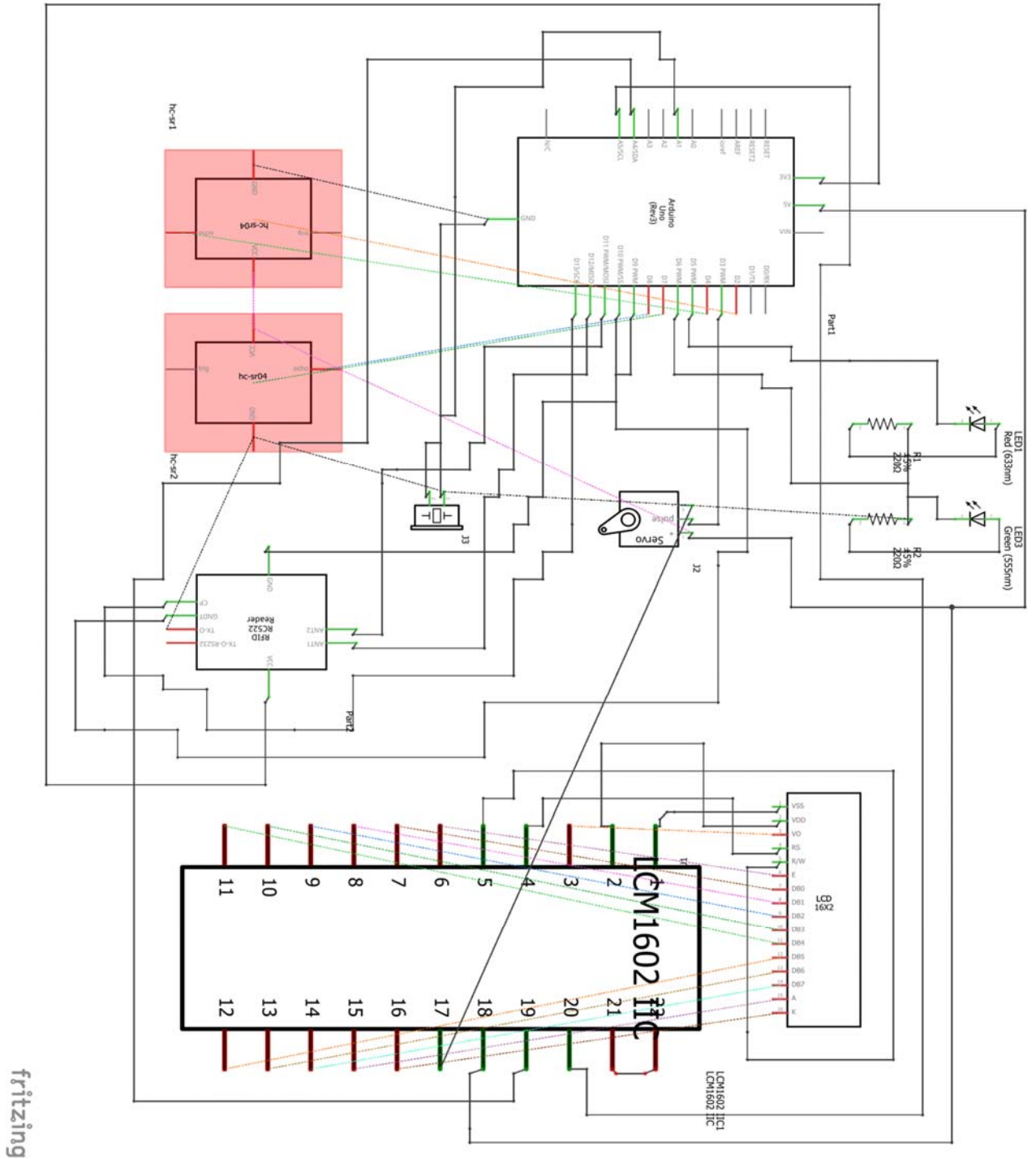


Figure 3.20: Schematic Diagram.

Prototype of the Toll System:

This image is captured by the side view of the prototype. Black and white colors is the road sign. We create our prototype by one-way road and manual toll payed booth is build left side of the design. All the sensors are put in this prototype by our system design. First we use ultrasonic no. one for counting wait time and the use RFID sensor for detect the user or car who has genuine RFID card access. Third one is the LCD display for that user can see the message of his or her next step. Left side is the manual portion of paid money if balance is low or any other problem occurred. There are two red and green lights for the indicator. Last two part is servo barrier and ultrasonic no. two which detect that the vehicles is gone or not.

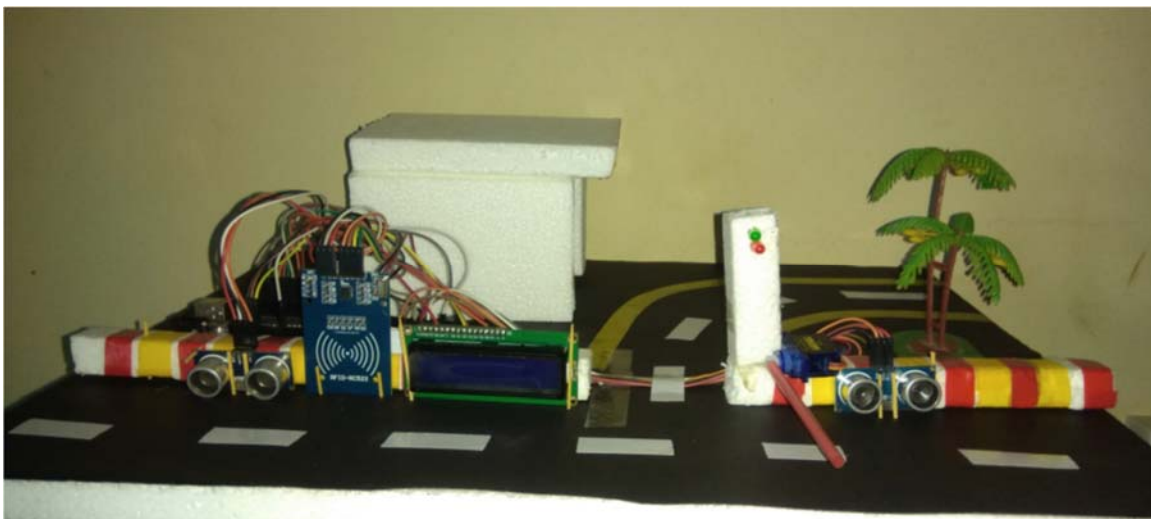


Figure 3.21: Prototype of the toll system.

Chapter 4

Description of The Software System

4.1 Introduction

In our project, we have used programming reference for the command structure and the basic syntax of the Arduino microcontroller and continues to describe the syntax of the most common elements of the languages and illustrates their usage with examples and code fragments which includes many functions of the core library of sample schematics and started programs. Written with the basic structure of Arduino's C & C++ derived programming language.

We have used Arduino Uno R3 which can be programmed using the Arduino's Desktop IDE software for offline use which can be downloaded from their website which supports Operating Systems of Windows, Mac OS X, Linux and Portable IDE (for Windows and Linux only), although Online IDE is available for use with internet connection. We have to select "Arduino Uno w/ATmega328" from the **Tools > Board** menu (microcontroller on our board). This ATmega328 on our used Arduino Uno R3 microcontroller comes with pre-burned with a boot-loader that allows us to upload new code to it without the use of an external hardware programmer.

4.2 Structure

The Arduino Language's basic structure is very simple and contains at least two parts or function. These functions enclose blocks of statements. These two functions are `setup()` and `loop()` function.

```
Void setup()
```

```
{  
    statements ;  
}
```

```
Void loop()
```

```
{  
    statements ;  
}
```

Both `setup()` and `loop()` functions are required for the program to run.

The `setup()` function contains preparation statements of the program and it contains declaration of variables. At the beginning of the program this function is run and for once after each power up or reset of the Arduino board and it is used to set pinMode or initialize the serial communication. This function initializes and sets the initial values.

The `loop()` function contains execution statements of the program and after running the `setup()` function, the `loop()` function is run and it contains the code which will be executed continuously like reading input, triggering output etc. This function allows our program to change and respond and to actively control the Arduino board. Actually `loop()` function is the core of all the Arduino program and it does the bulk of the work.

4.3 Variable Declaration

Variable is used for naming and storing a value for use it later by the program. Before use variables, we need to declare this variables that means defining its type and optionally assign its initial value.

This declaration needs only once at a time in a program but later value can be changed using arithmetic and others assignments. It can be declare in any location throughout the program. [14]

Some Variable types are:

- int
- char
- float
- double
- byte
- unsigned int
- long
- unsigned long

Example:

```
int scanVariable = 4;           //initialize the scan Variable as integer value 4
```

4.4 Arithmetic Operator

By using Arithmetic Operators, we can perform addition, subtraction, multiplication and division of operands. We also get remainder and the results are saved into variables using assignment operator.

Arithmetic Operators are:

- Assignment Operator (=)
- Addition Operator (+)
- Subtraction Operator (-)
- Multiplication Operator (*)
- Division Operator (/)
- Modulo Operator (%)

Examples:

$x = a + 7;$

$y = b - 5;$

$z = c * 3;$

$m = d / 6;$

$n = e \% 5;$

4.5 Compound Operator

By using Compound Operators, we can perform increment, decrement, compound addition, compound subtraction, compound multiplication, compound division, compound modulo, compound bitwise AND, compound bitwise OR etc. Actually this operator combines arithmetic operation with variable assignment and most commonly used on loop operations.

Arithmetic Operators are:

- Increment Operator (++)
- Decrement Operator (--)
- Compound Addition (+=)

- Compound Subtraction (-=)
- Compound Multiplication (*=)
- Compound Division (/=)
- Compound Modulo (%=)
- Compound Bitwise **AND** (&=)
- Compound Bitwise **OR** (|=)

Examples:

```
a++;          // increment a by 1
b--;          // decrement b by 1
c += x;       // increment c by +x
d -= y;       // decrement d by -y
x *= y;       // multiple x by y and save to x
x /= y;       //divide x by y and save to x
```

4.6 Boolean Operator

By using Boolean Operator, we can compare two expressions and get Boolean value and it return a Boolean value of 0 or 1. If the value is TRUE, it returns 1 and if the value is FALSE, it returns 0. This operators are frequently used inside the condition of an if statement.

Boolean Operators are:

- Logical AND (&&)
- Logical OR (||)
- Logical Not (!)

Examples:

```
if(x>2 && x<9) { ... }      // if only both expression are true, it will return TRUE
```

```
if(x>2 || y>2) { ... }     // if any expression is true, it will return TRUE
```

```
if(!x) { ... }             // if operand is false, it will return TRUE
```

4.7 pinMode(pin,mode)

Digital I/O function pinMode are used in void setup() to configure the specified pin to behave either as an INPUT or as an OUTPUT.

Its syntax is pinMode(pin, mode), here, the two parameters here pin's value will be the number of pin whose mode we wish to set and mode's value will be INPUT, OUTPUT or INPUT_PULLUP and this function will not return anything.[15]

Example:

```
int ledPin = 13;           // LED connected to digital pin 13

void setup()
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
}
```

```
digitalWrite(ledPin, LOW); // sets the LED off
delay(1000);              // waits for a second
}
```

4.8 digitalWrite(pin, Value)

digitalWrite() function writes a HIGH or LOW value to a specified digital pin and this pin can be specified as a variable or constant. Its syntax is written as digitalWrite(pin, value), where the parameter pin is for the pin number and value can be either HIGH or LOW.[16]

Example:

```
int ledPin = 13;          // LED connected to digital pin 13

void setup(
{
pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop()
{
digitalWrite(ledPin, HIGH); // sets the LED on
delay(1000);                // waits for a second
digitalWrite(ledPin, LOW);  // sets the LED off
delay(1000);                // waits for a second
}
```

In this code, sets pin 13 to HIGH, makes a one-second-long delay, and sets the pin back to LOW.

4.9 Delay(ms)

The Delay() function actually pauses the program for the millisecond amount of time. Its syntax is written as delay(ms). Its parameter is ms means millisecond is the value of number of milliseconds to pause. there are 1000 millisecond in a second.

Example:

```
delay(1000);          // pause for 1 second (1000 millisecond)
```

4.10 Serial.begin(speed)

Serial.begin() function is used to open serial port and set the data rate in bits per second (baud) for serial data transmission. The typical baud rate for communicating with the computer is 9600 although other speeds are supported like 300, 600, 1200, 2400, 4800, 14400, 19200, 28800, 38400, 57600, or 115200 etc. When using serial communication, digital pins 0(RX) and 1(TX) cannot be used at the same time.

Its Syntax is Serial.begin(speed). speed parameter is value in bits per second (baud).

Optional config parameter can be used for set data, parity, and stop bits.

Example:

```
void setup() {  
  
    Serial.begin(9600);          // open serial port and sets data rate to 9600 bps  
  
}
```

4.11 Serial.println(val)

Serial.println() function is used to print data to the serial port as human readable ASCII text followed by a carriage return character and a newline character. This command takes the same forms as Serial.print() but it is easier for reading data serial monitor.

Its syntax is written as Serial.println(val), where the parameter val can be the value to print of any data type. Optional parameter format can be used for specify the number base for integer data types and number of decimal places for floating point types. [17]

Example:

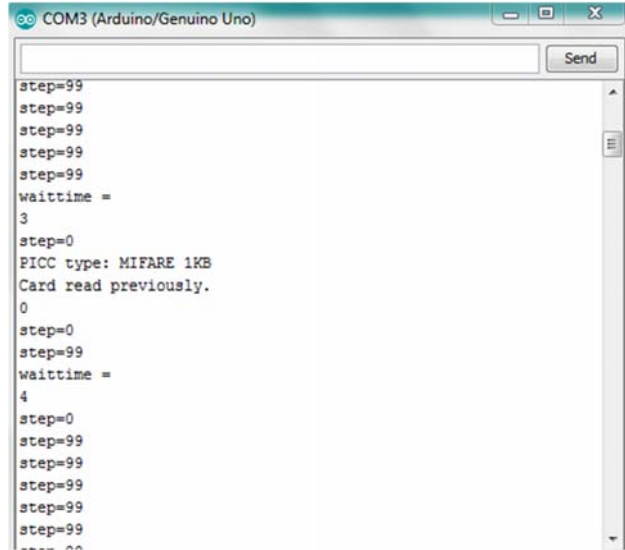
```
int analogValue = 0; // variable to hold the analog value

void setup() {
    Serial.begin(9600); // open the serial port at 9600 bps
}

void loop() {
    analogValue = analogRead(0);           // read the analog input on pin 0
    // print it out in many formats
    Serial.println(analogValue);           // print as an ASCII-encoded decimal
    Serial.println(analogValue, DEC);      // print as an ASCII-encoded decimal
    Serial.println(analogValue, HEX);      // print as an ASCII-encoded hexadecimal
    Serial.println(analogValue, OCT);      // print as an ASCII-encoded octal

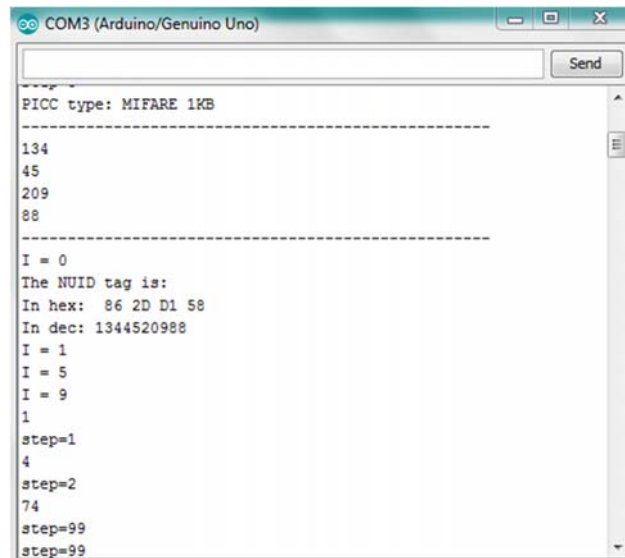
    Serial.println(analogValue, BIN);      // print as an ASCII-encoded binary
    delay(10);                             // delay 10 milliseconds before the next reading
}
```

Serial monitor Output:



```
COM3 (Arduino/Genuino Uno)
step=99
step=99
step=99
step=99
step=99
waittime =
3
step=0
PICC type: MIFARE 1KB
Card read previously.
0
step=0
step=99
waittime =
4
step=0
step=99
step=99
step=99
step=99
step=99
step=99
step=99
step=99
```

Figure 4.1: Output of serial monitor 1



```
COM3 (Arduino/Genuino Uno)
PICC type: MIFARE 1KB
-----
134
45
209
88
-----
I = 0
The NUID tag is:
In hex: 86 2D D1 58
In dec: 1344520988
I = 1
I = 5
I = 9
1
step=1
4
step=2
74
step=99
step=99
```

Figure 4.2: Output of serial monitor 2

Chapter 5

Conclusion

We have successfully implemented our method on the proposed automated toll collection system. Our toll collection system is very fast and efficient mode for collection toll. This saves a lot of time since vehicles passing through the toll system do not stop to pay toll and the payment automatically takes place from the account of the vehicle. It is observed that there is reduction in the number of accident caused near the toll plazas due to considerable decrement in congestion around toll system. Since the vehicles do not stop at the toll facility. This has reduced the congestion of the toll system nicely.

This toll collection system is mainly based on RFID, a design scheme is given hardware section. The whole system is developed to faster the toll collection system and ensure security in highways and bridges. Usage of RFID technology ensures the system as a secure system as RFID holds a unique identification number and there is no chance of cloning. It is low cost, high security, far communication and efficiency, etc. This toll collection system using RFID is an effective measure to reduce management costs and fees, at the same time, greatly reduce noise and pollutant emission of toll station. In the design of the proposed toll system, real life toll collection, anti-theft or without any RIFD card user car is caught automatically at the toll booth so this system has been designed. This system of collecting tolls is ecofriendly and also results in increased toll lane capacity. Also an anti-theft solution system module which prevents passing of any defaulter vehicle is implemented, thus assuring security on the roadways. On the whole, the system is hassle free and user friendly.

5.1 Future Work

In our Future improvement we use high speed image capture camera so that any car can't bypass our system or doing any unethical activity on road. We also try to make our system more accurate. Also we will be probably implementing the real time database which is connect with government created roads and highway database server. When people will frequently use RFID, then we can add a feature like if a car hasn't sufficient balance on its account, he can use it but the toll money will deduct from his next recharge, so the necessity of our manual toll booth will no longer available in future.

References

- 1) <https://www.arduino.cc/en/Guide/Introduction>
- 2) <https://www.arduino.cc>
- 3) <https://en.wikipedia.org/wiki/Arduino#History>
- 4) <https://playground.arduino.cc/Learning/MFRC522>
- 5) <https://www.techopedia.com/definition/26992/radio-frequency-identification-reader-rfid-reader>
- 6) https://en.wikipedia.org/wiki/Radio-frequency_identification#History
- 7) <https://www.google.com.bd/#q=servo+motor+datasheet>
- 8) <http://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html>
- 9) <https://www.arduino.cc/en/Tutorial/HelloWorld>
- 10) <http://modtronix.com/mod-lcdi2c-bb1.html>
- 11) <https://www.sparkfun.com/products/13959>
- 12) www.micropik.com/PDF/HCSR04.pdf
- 13) <https://en.wikipedia.org/wiki/Buzzer>
- 14) <https://www.arduino.cc/en/Reference/VariableDeclaration>

15) <https://www.arduino.cc/en/Reference/PinMode>

16) <https://www.arduino.cc/en/Reference/DigitalWrite>

17) <https://www.arduino.cc/en/Serial/Println>

18) <http://bdnews24.com/bangladesh/2017/06/21/gridlocks-on-kanchpur-meghna-bridges-feared-during-eid-travel>

19) <https://espace.curtin.edu.au/bitstream/handle/20.500.11937/52601/251220.pdf?sequence=2&isAllowed=y>

20) <http://www.bdwave24.com/?p=1884>

Appendix

Code

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Ultrasonic.h>

Ultrasonic ultrasonic(7,8);
Ultrasonic ultrasonic0(2,4);
LiquidCrystal_I2C lcd(0x3F,16,2); // set the LCD address to 0x3F for
a 16 chars and 2 line display
#define SS_PIN 10
#define RST_PIN 9
#define Red_Btn 5
#define Green_Btn 6
#define roadwidth 5
#define alarm A1

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class

MFRC522::MIFARE_Key key;
Servo myservo;
int pos = 0;
int waittime = 0;
int codeswitch = 99;
// Init array that will store new NUID
byte nuidPICC[4];
int rfidarray[] = {134,45,209,88,102,149,180,88,182,85,176,88};
int taka[] = {300,0,450};
int personidentifier;

void setup() {
  Serial.begin(9600);
  pinMode(Green_Btn, OUTPUT);
  pinMode(Red_Btn, OUTPUT);
  lcd.init(); // initialize the lcd
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522
  myservo.attach(3);
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}
```

```

Serial.println(F("This code scan the MIFARE Classsic NUID.));
Serial.print(F("Using the following key:"));
printHex(key.keyByte, MFRC522::MF_KEY_SIZE);
}

void loop() {
if(codeswitch == 99)
{
digitalWrite(Red_Btn, HIGH);
Serial.println("step=99");
if(ultrasonic0.distanceRead(<roadwidth)
{
if(waittime>=10)
{ lcd.clear();
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("Please Go Left ^");
lcd.setCursor(0,1);
lcd.print("_____|");
while(ultrasonic0.distanceRead(<roadwidth){
tone(alarm,2000,500); //tone(pin, frequency, duration)
delay(1000);
Serial.println("Inside loop");
}
noTone(alarm);
waittime = 0;
lcd.clear();
}
else{
waittime = waittime + 1;
Serial.println("waittime = ");
Serial.println(waittime);
delay(1000);
codeswitch = 0;
}
}
}
if(codeswitch == 0)
{
//delay(200);
Serial.println("step=0");
digitalWrite(Red_Btn, HIGH);
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("-----STOP-----");
// Look for new cards
if ( ! rfid.PICC_IsNewCardPresent()){
codeswitch = 99;
return;
}
}

// Verify if the NUID has been readed

```

```

if ( ! rfid.PICC_ReadCardSerial())
return;

Serial.print(F("PICC type: "));
MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
Serial.println(rfid.PICC_GetTypeName(piccType));

// Check is the PICC of Classic MIFARE type
if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
Serial.println(F("Your tag is not of type MIFARE Classic."));
return;
}

if (rfid.uid.uidByte[0] != nuidPICC[0] ||
rfid.uid.uidByte[1] != nuidPICC[1] ||
rfid.uid.uidByte[2] != nuidPICC[2] ||
rfid.uid.uidByte[3] != nuidPICC[3] ) {
Serial.println("-----");
Serial.println(rfid.uid.uidByte[0]);
Serial.println(rfid.uid.uidByte[1]);
Serial.println(rfid.uid.uidByte[2]);
Serial.println(rfid.uid.uidByte[3]);
Serial.println("-----");
int j = 0;
for (byte i = 0; i < 12 ; i++) { // 12 mean 3 Id 4 + 4 + 4
Serial.print("I = ");
Serial.println(i);
if(rfid.uid.uidByte[j] == rfidarray[i] && rfid.uid.uidByte[j+1] ==
rfidarray[i+1] && rfid.uid.uidByte[j+2] == rfidarray[i+2] &&
rfid.uid.uidByte[j+3] == rfidarray[i+3])
{
if(i==0){
personidentifier = 0;
}
else{
personidentifier = i/4;
}
if(taka[personidentifier]>49){
taka[personidentifier] = taka[personidentifier] - 50;
lcd.clear();
lcd.backlight();
lcd.setCursor(2,0);
lcd.print("Person No:=>");
lcd.print(personidentifier+1);
lcd.setCursor(2,1);
lcd.print("Balance:=");
lcd.print(taka[personidentifier]);
delay(3000);
codeswitch = 1;
lcd.clear();
}
}
}
}

```

```

lcd.backlight();
lcd.setCursor(3,0);
lcd.print("Please Go");
digitalWrite(Green_Btn, HIGH);
digitalWrite(Red_Btn, LOW);
for (pos = 0; pos <= 90; pos += 1) { // goes from 0 degrees to 90
degrees
// in steps of 1 degree
myservo.write(pos);           // tell servo to go to position in
variable 'pos'
delay(15);                     // waits 15ms for the servo to reach
the position
}
}
else{
lcd.clear();
lcd.backlight();
lcd.setCursor(2,0);
lcd.print("Balance Low");
lcd.setCursor(2,1);
lcd.print("Please Go Left");
delay(4000);
lcd.clear();
waittime = 0;
}

// Store NUID into nuidPICC array
for (byte i = 0; i < 4; i++) {
nuidPICC[i] = rfid.uid.uidByte[i];
}

Serial.println(F("The NUID tag is:"));
Serial.print(F("In hex: "));
printHex(rfid.uid.uidByte, rfid.uid.size);
Serial.println();
Serial.print(F("In dec: "));
printDec(rfid.uid.uidByte, rfid.uid.size);
Serial.println();
}
else{
i = i+3;
}
}
}
else
{
lcd.clear();
Serial.println(F("Card read previously.));
lcd.backlight();
lcd.setCursor(3,0);
lcd.print("Card Read");
lcd.setCursor(3,1);

```

```

lcd.print("Previously.");
delay(1000);
// clear the screen
lcd.clear();
}
Serial.println(codeswitch);
// Halt PICC
rfid.PICC_HaltA();

// Stop encryption on PCD
rfid.PCD_StopCryptol();
}
else if(codeswitch == 1)
{
waittime = 0;
delay(300);
Serial.println("step=1");
Serial.println(ultrasonic.distanceRead());
if(ultrasonic.distanceRead()<roadwidth)
{
codeswitch = 2;
}
}
else if(codeswitch == 2)
{
delay(100);
Serial.println("step=2");
Serial.println(ultrasonic.distanceRead());
delay(100);
if(ultrasonic.distanceRead()>roadwidth)
{
digitalWrite(Green_Btn, LOW);
digitalWrite(Red_Btn, HIGH);
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("-----STOP-----");
for (pos = 90; pos >= 0; pos -= 1) { // goes from 90 degrees to 0
degrees
myservo.write(pos);           // tell servo to go to position in
variable 'pos'
delay(15);                     // waits 15ms for the servo to reach
the position
}
codeswitch = 99;
}
}
}
}
/**
 * Helper routine to dump a byte array as hex values to Serial.
 */
void printHex(byte *buffer, byte bufferSize) {
for (byte i = 0; i < bufferSize; i++) {

```

```
Serial.print(buffer[i] < 0x10 ? " 0" : " ");
Serial.print(buffer[i], HEX);
}
}
/**
 * Helper routine to dump a byte array as dec values to Serial.
 */
void printDec(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], DEC);
  }
}
```