PROJECT REPORT

ON

MODERN UNIVERSITY

Supervised by

Dr. Moe Thuzar Htwe

Professor

Faculty of Computer Systems and Technologies

University of Computer Studies (Magway)

Submitted by

Group-A

4CT-1	Ma Shwe Yamone Oo
4CT-5	Ma Phoo Theingi Soe
4CT-6	Ma Mya Wai Kyaw
4CT-8	Mg Wai Phyo Aung
4CT-16	Mg Paing Thet Kyaw
4CT-24	Ma Akari Phyoe

2022-2023

ACADEMIC YEAR

ACKNOWLEDGEMENTS

First, we would like to thank Dr. Soe Lin Aung, Pro-Rector of the University of Computer Studies (Magway) for his helpful guidelines on this project.

Secondly, we are extremely grateful to our supervisor Dr. Moe Thuzar Htwe, Professor and Head of Faculty of Computer Systems and Technologies, and other teachers for reviewing our project presentation.

We would like to express great appreciation to Daw Chaw Su Hlaing, Associate Professor and Head of English Department for reviewing the project from the language point of view.

ABSTRACT

Our project is automatic technology, to record the attendance of students in modern universities, to maintain the security of the machine rooms, to make it easy to get in and out of the car. Using a application based on IoT, the water level in the water reservoirs in the campus and the university can be easily viewed from the application to turn on and off the lights remotely. As modern times progress, universities have student classes. There is no need to count attendance and absences, and by using RFID-based entry and exit cards, it is easy to detect absences. Therefore, it saves time and helps to check roll call incompleteness. It also saves time because it includes door systems that automatically open and close. If there is a fire or smoke is detected, an alarm sounds and sprinkles water to extinguish the fire. The unique feature is that wheelchair users can easily attend Stairs have been installed. Modern universities play a crucial role in shaping the future by providing access to higher education, fostering innovation, and conducting valuable research.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

We made this project with the aim of building better universities in the future. In this project, an application that saves time for human resources and can be controlled from anywhere is written. In an era characterized by rapid technological advancements and a growing emphasis on safety and security, universities play a pivotal role in ensuring the well-being of their students, staff, and resources. As educational institutions become more complex and interconnected, the need for modern, efficient, and intelligent security systems has never been greater. It has a fire alarm system; people can be safe because it can be detected in advance when a fire occurs. In the Machine Room system, it is controlled by username/password, so it improves security. The aim of this embedded project is to create a seamless and intelligent security ecosystem that leverages cutting-edge technology. This system will not only ensure the safety of the campus but also enhance the overall efficiency and management of the university's resources.

1.1 Objective of the Project

- ⇒ To automatically open the main door and car parking door when an object comes.
- ⇒ To control water fountains, light bulbs and water level measurements with the application.
- ⇒ To give an alarm and open the door, if there is a fire and then to spray water from the water pump.
- ⇒ To check the attendance of the student with RFID card system.
- ⇒ To apply embedded system in real world for communication
- ⇒ To know Design and build a reliable embedded system that encompasses fire detection, access control, attendance monitoring, and IoT communication capabilities.
- ⇒ To Ensure the embedded system's architecture is scalable to accommodate future expansions or additional security features without significant engineering.

CHAPTER 2

HISTORY OF EMBEDDED SYSTEM

2.1 Background Theory

An embedded system is a microprocessor-based hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system. At the core is an integrated circuit designed to carry out computation for real-time operations.

The complexity of an embedded system varies significantly depending on the task for which it is designed. Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems.

Milestones in the History of Embedded Systems:

1950s-1960s - Emergence of Early Embedded Systems

1970s - Microcontrollers and Microprocessors

1980s-1990s - Embedded Systems in Consumer Electronics

1990s-Present - Embedded Systems in the Internet of Things (IoT)

2000s-Present - Advances in Processing Power

Embedded system applications range from digital watches and microwaves to hybrid vehicles and avionics. As much as 98 percent of all microprocessors manufactured are used in embedded systems.

1. Stand-alone Embedded Systems

2. Real-time Embedded Systems

   (i) Soft Embedded Systems

   (ii) Hard Embedded Systems

3. Network Embedded Systems

4. Mobile Embedded Systems

2.2 Type of Processor in Embedded system

Microcontrollers (MCUs): These are small, single-chip processors with integrated memory and peripherals. MCUs are commonly used in applications with low power requirements and limited computational needs. Examples include Arduino's AT mega series, PIC microcontrollers, and ARM Cortex-M series.

Microprocessors (MPUs): Unlike MCUs, microprocessors do not have integrated memory and peripherals, requiring external components. They are used in more complex applications where computational power is a priority. Examples include the Intel x86 processors, ARM Cortex-A series, and MIPS processors.

Digital Signal Processors (DSPs): DSPs are optimized for tasks involving signal processing, such as audio and video processing, image recognition, and telecommunications. They excel at handling mathematical operations required for these applications.

Field-Programmable Gate Arrays (FPGAs): FPGAs are reconfigurable hardware devices that can be programmed to implement custom digital logic circuits. They are used in applications that require high-speed and highly parallel processing, such as aerospace, telecommunications, and industrial automation.

System-on-Chip (SoC): SoCs integrate various components, including processors, memory, input/output interfaces, and often GPU or FPGA elements, into a single chip. They are prevalent in applications like smartphones, tablets, and embedded systems that require a high level of integration.

CHAPTER 3

HARDWARE REQUIREMENTS AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

The following components are required to create **MODERN UNIVERSITY.** They are :-

1.Microcontroller(ATMega328P)

2.RFID Reader and Tag(MFRC522)

3.Liguid Crystal Display(20*4)&(16*2)

4.Keypad(4*4)

5.Servo Motor(SG 90)

6.Buzzer

7.Light Emitting Diode(LED)

8.Jumper Wires

9.Male/Female Jack Connector

10.AC/DC Adapter(9V)

11.IR Sensor(Infrared Sensor)

12.DC Pump(5V)

13.Ultrasonic Senor(HC-SR04)

14.Fame Sensor(RU2336573C1)

15.Relay(3 Channel)

16.Node MCU(ESP82660&(ESP32)

3.1.1 Microcontroller (ATMega328P)

A microcontroller is a compact integrated circuit designed to perform a specific operation in an embedded system. There are many microcontrollers. Among them, this system was implemented using Arduino Uno.

Arduino Uno used in this system is based on the ATMega328P microcontroller. It has 14 digital I/O pins, 6 analog pins, a 16 MHz crystal oscillator, a USB port, a power jack, an ICSP header and a reset button. Arduino uno includes 3 types of memory, Flash memory, SRAM and EEPROM. Flash memory is used to store program images and any initialized data. RAM or Static Random Access Memory can be read and written from the executing program. RAM memory is used for several purposes

by a running program: Static Data, Heap and Stack. EEPROM is another form of non-volatile memory that can be read or written from executing program.

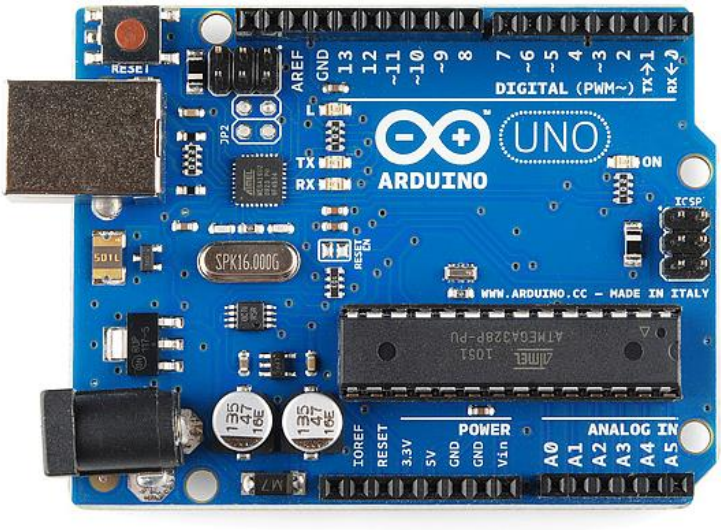


Figure 3.1: Arduino Uno Pinout

3.1.2 RFID Reader and Tag (MFRC522)

RFID is a form of wireless communication that incorporates the use of electromagnetic in the radio frequency portion of the electromagnetic spectrum. Every RFID system consists of three components: a scanning antenna, a transceiver, and a transponder. An RFID reader is from the combination of an antenna and a transceiver. The transponder is in the RFID tag itself. The read range for RFID tags varies based on factors including the type of tag, type of reader, RFID frequency and interference in the surrounding environment or from other RFID tags and readers.

RFID tags are usually identified by their radio frequencies: low frequency (LF), high frequency (HF), and ultra-high frequency (UHF). LF systems have a range between 30 and 500 MHz (typically 125KHz) and a read range less than 3 feet. HF systems have a range between 3 and 30 MHz (typically 13.56MHz) and a read range less than 6 feet. UHF systems have a range between 300 MHz and 3 GHz and a read range up to 12 m (39 ft).

RFID module used in this system is MFRC522 MI Fare classic 1K Module which has a read range of around 5 cm, which is shown in figure 3.2.

Figure 3.2: RFID Reader and Tag

3.1.3 Liquid Crystal Display (LCD)

We can easily interface a liquid crystal display (LCD) with an Arduino to provide a user interface. Liquid crystal displays (LCDs) are commonly used to display data in devices such as calculators, microwave ovens, and many other electronic devices.

A 20x4 LCD means it can display 20 characters per line and there are 4 such lines. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. Each character is displayed in a 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

In the state in which molecules are in a uniform direction, they also have refractve indices, dielectric constants and other physical characteristics similar to those of crystals, depending on their direction, even though they are liquid. This is why they are called liquid crystal.

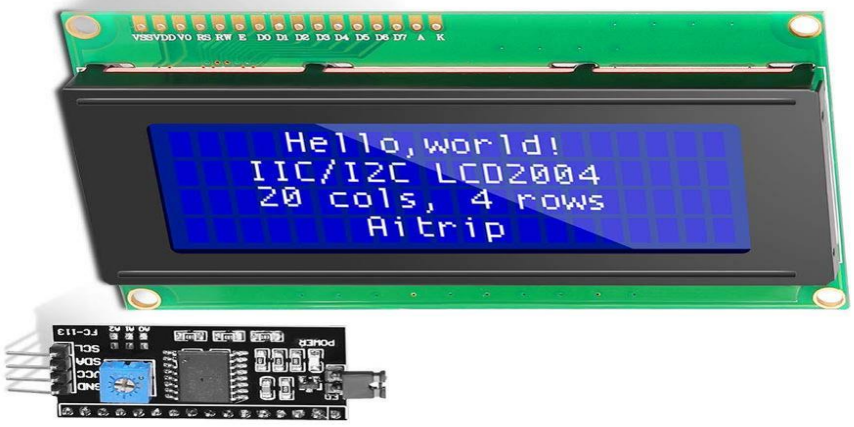


Figure 3.3: Liquid Crystal Display

3.1.4 Keypad (4*4 keypad)

Keypad is used as an input device to read the key pressed by the user and to process it.4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns. A key press establishes a connection between the corresponding row and column, between which the switch is placed. 4x4 keypad is shown in fig 3.4.

Figure 3.4: Keypad (4x4)

3.1.5 Servo Motor (SG-90)

Servo motors are great devices that can turn to a specified position. Usually, they have a servo arm that can turn 180 degrees. Using the Arduino, we can tell a servo to go to a specified position and it will go there.

Figure 3.5: Servo Motor

3.1.6 Buzzer

An Arduino Buzzer is basically a beeper. The Arduino buzzer is a device that produces sound when an electric current is passed through it. The Arduino buzzer can be directly connected to the Arduino and produces different tones by giving different frequency electric pulses to the buzzer.

Figure 3.6: Buzzer

3.1.7 Light Emitting Diode (LED)

A Light Emitting Diode (LED) is a semiconductor device, which can emit light when an electric current passes through ti. To do this, holes from p-type semiconductors recombine with electrons from n-type semiconductors to produce light. Various types of LEDs are shown in fig 3.7.

Figure 3.7: Light Emitting Diode

3.1.8 Jumper Wires

A jumper wire is an electric wire that connects remote electric circuits used for printed circuit boards. By attaching a jumper wire on the circuit, it can be short-circuited and short-cut (jump to the electric circuit). Jumper wires are shown in fig 3.8.

Figure 3.8: Jumper wires

3.1.9 Male/Female Jack Connector

A "male" connector usually has a pin or pins and the "female" connector is designed to receive those pins. Male and Female connectors are shown in fig 3.9.

Figure 3.9: Male/Female Jack Connector

3.1.10 AC/DC Adapter (9V)

AC/DC adapters (9V) are used as power supply for controller and others peripheral devices.AC/DC adapter (9V) is shown in fig 3.10.

Figure 3.10: AC/DC Adapter (9V)

3.1.11 IR Sensor

The IR sensor module includes five essential parts like IR Tx, Rx, Operational, amplifier, trimmer pot (variable resistor) & output LED. The pin configuration of the IR sensor module is discussed below.

IR PROXIMITY SENSOR

VCC Pin is a power supply input.

GND Pin is power supply ground.

OUT is an active-high o/p.

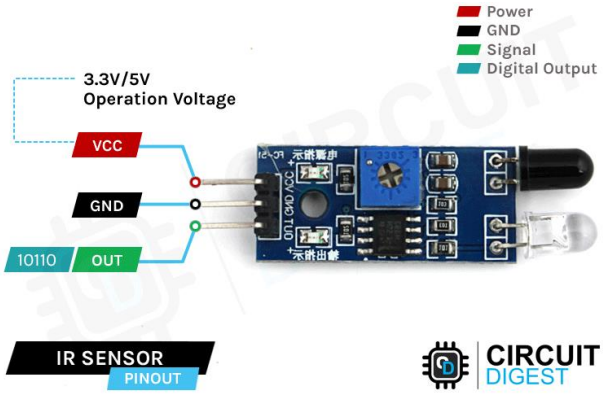


Figure 3.11: IR sensor

3.1.12 DC Pump (5V)

This is a low cost, small size Submersible Pump Motor which can be operated from a 2.5 ~ 6V power supply. It can take up to 120 litres per hour with very low current consumption of 220mA.Just connect tube pipe to the motor outlet, submerge it in water and power it.

Figure 3.12: DC water pump

3.1.13 Ultrasonic Sensor

Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by Distance = Speed/170.15 m × Meters\100 cm × 1e6 µS/170.15 m × 58.772 µS/cm**.**

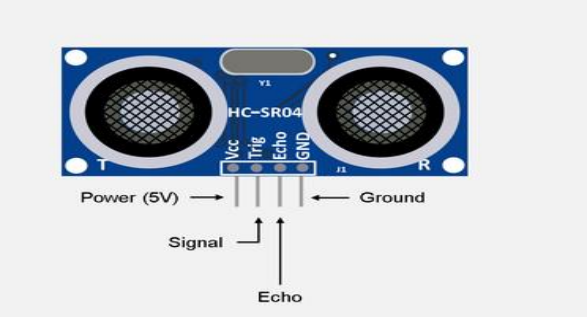


Figure 3.13: Ultrasonic Sensor

3.1.14 Flame Sensor

A flame sensor is a crucial safety component on your gas heating system. During the ignition cycle, your gas furnace goes through a process where a spark or a hot surface ignitor will ignite the gas. As the gas is ignited, the flame sensor creates a current of electricity. The electricity is calculated in micro amps. If the furnace's control board does not read the proper level of micro amps, the furnace will no longer give the system fuel to avoid an explosion.

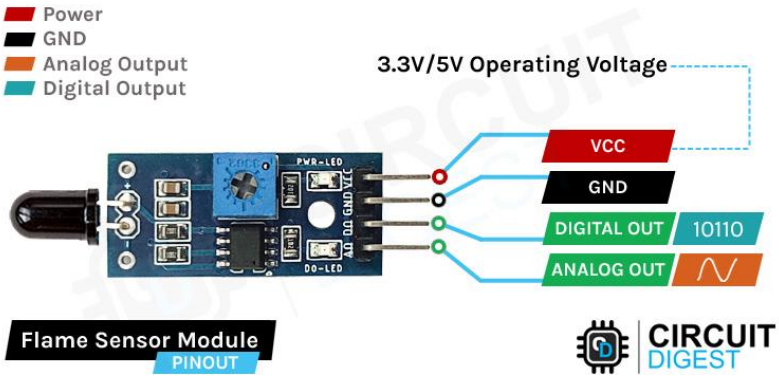


Figure 3.14: Flame Sensor

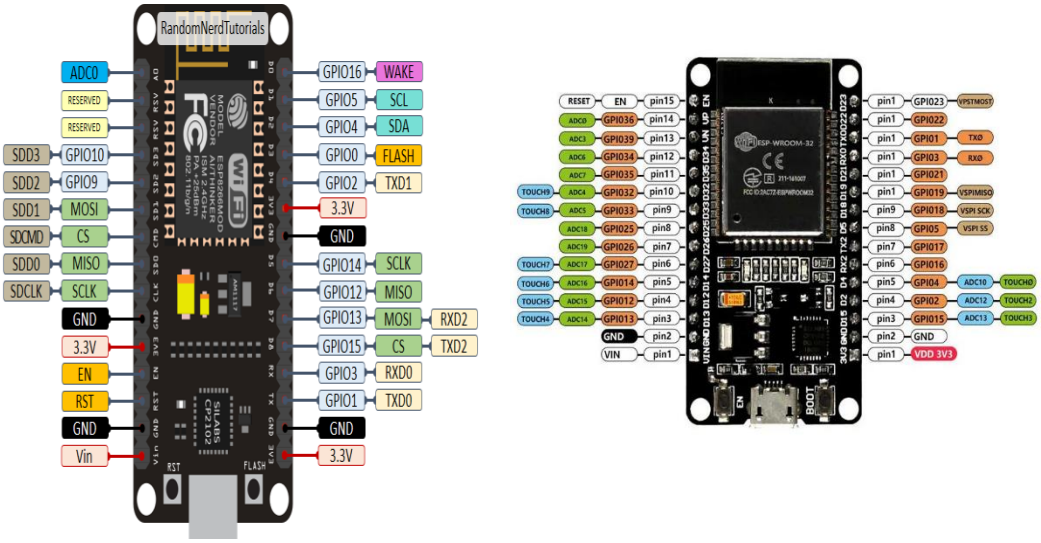
3.1.15 Relay (3-Channel)

A relay is an electrical switch that can be used to control devices and systems that use higher voltages. In the case of module relay, the mechanism is typically an electromagnet. The relay module input voltage is usually DC. However, the electrical load that a relay will control can be either AC or DC, but essentially within the limit levels that the relay is designed for. A relay module is available in an array of input voltage ratings: It can be a 3.2V or 5V relay module for low power switching, or it can be a 12 or 24V relay module for heavy-duty systems. The relay module information is normally printed on the surface of the device for ready reference. This includes the input voltage rating, switch voltage, and current limit.

Figure 3.15: 3-Channel Relay

3.1.15 Node MCU

The ESP8266 uses a 32bit processor with 16-bit instructions. It is Harvard architecture which mostly means that instruction memory and data memory are separate. The Node MCU (Node Microcontroller Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SOC) called the ESP8266. The ESP8266, designed and manufactured by Express if Systems, contains the crucial elements of a computer: CPU, RAM, networking (Wi-Fi), and even a modern operating system and SDK. That makes it an excellent choice for Internet of Things (IoT) projects of all kinds.



ESP-8266 ESP-32

Figure 3.16: Node MCU

3.2 Software Requirements

Microcontrollers need to write programs to perform specific tasks. In this system, Arduino IDE for program Arduino board and Node MCU 32. As for financial management android applications, MIT app inventor version 1 is used.

3.2.1 Arduino IDE

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

Arduino code is written in **C++** with an addition of special methods and functions, which will be mentioned later. **C++** is a human-readable programming language. When we create a 'sketch' (the name given to Arduino code files), it is processed and compiled to machine language.

Figure 3.17: Arduino IDE Logo

3.2.2 MIT APP INVENTOR

MIT App Inventor is an intuitive, visual programming environment that allows everyone, even children, to build fully functional apps for smartphones and tablets. App Inventor lets us develop applications for android phones using a web browser and either a connected phone emulator. The App Inventor servers store we work and help we keep track of our projects. We build apps by working with: The App Inventor Designer, where you select the components for our app. Figure 3.18 is the logo of MIT app inventor.

Figure 3.18: MIT App Inventor Logo

CHAPTER 4

DEDIGN AND IMPLEMENTATION

4.1 Design of Modern University

Our Modern University System includes “Open Door System”, “Car parking System”, “Fire Alarm System”, “Machine Door System”, ‘Attendance System”, “IoT Control App System”.

Figure 4.1: Design of Modern University

4.1.1 Fire Alarm System

A fire alarm system is a crucial part of the fire and life safety of a building and its occupants. It is designed to detect and alert occupants and emergency forces of the presence of smoke, fire or other fire-related emergencies. The system is required in most commercial buildings.

4.1.2 Block Diagram for Fire Alarm System

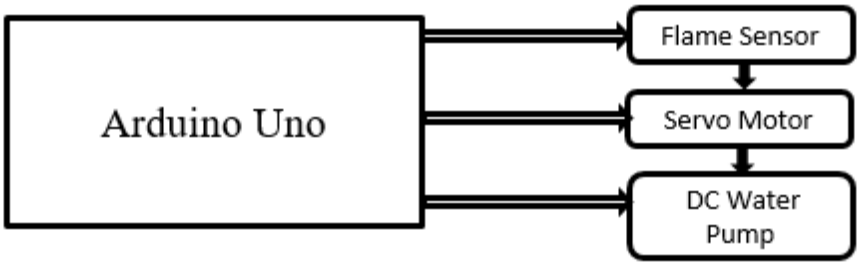


Figure 4.2: Block Diagram of Fire Alarm System

4.1.3 Circuit Diagram for Fire Alarm System

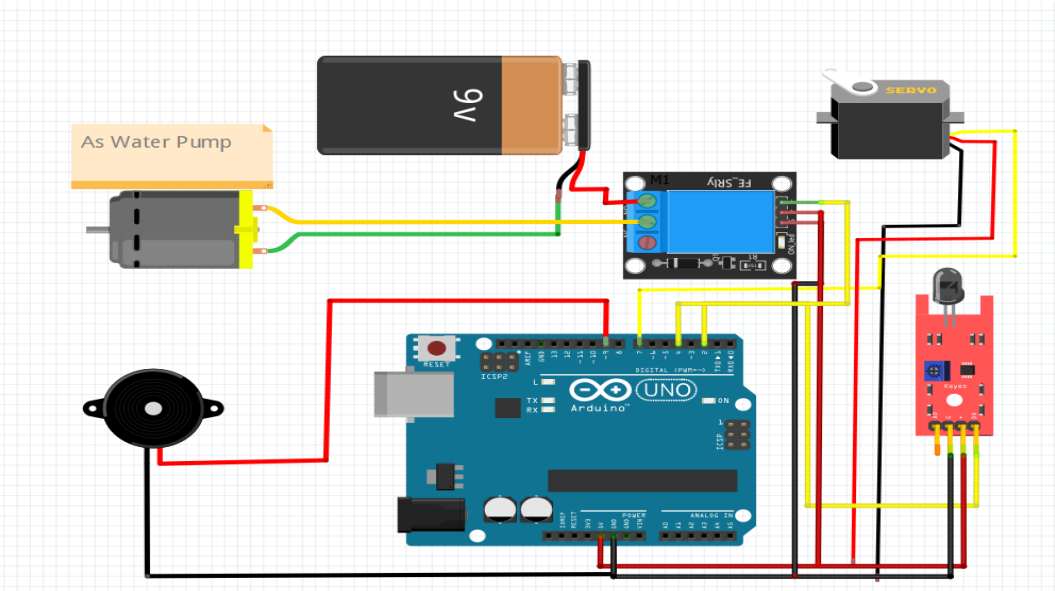


Figure 4.3: Circuit diagram for Fire Alarm System

4.1.4 System Flow Diagram for Fire Alarm System

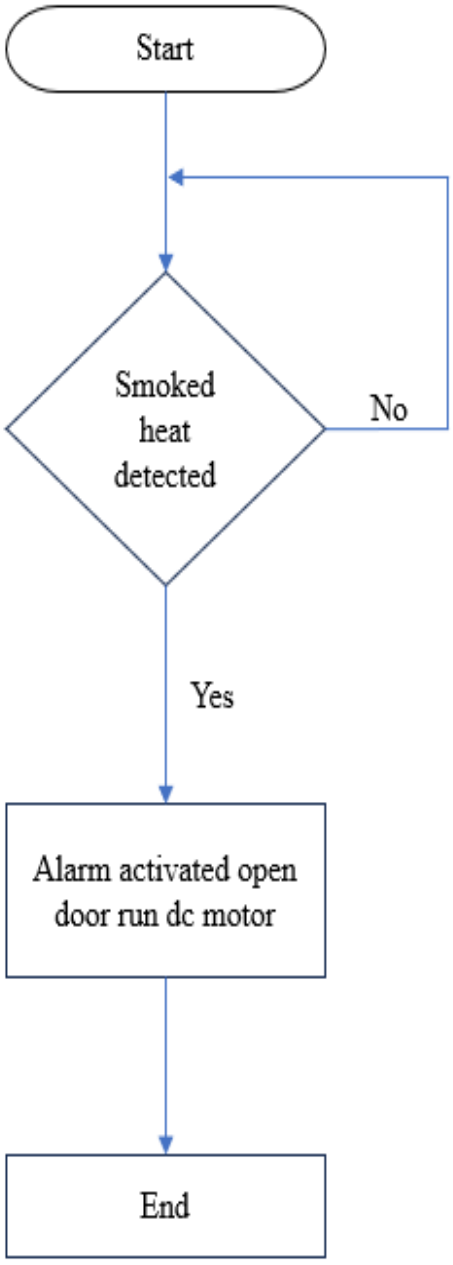


Figure 4.4: Flow Diagram for Fire Alarm System

No	Device's Name	Device's Pins	Arduino Uno's Pin Number
1	Flame Sensor	A0 Pin GND Pin VCC Pin D0 Pin	Pin Not Used. GND Pin 5V Pin D2
2	Buzzer	VCC GND	D9 Pin GND
3	Servo Motor	Control Signal VCC GND	D7 5V Pin GND
4	5V DC Water Pump	Positive Negative	Common Battery's Negative
5	9V Battery	Positive Negative	Common DC's Negative
6	2-Channel relay	Common Normally Closed	Battery's Positive DC Pump's Positive

Table 4.1: Fire Alarm Pin Assignment Table

4.1.5 Implementation for Fire Alarm System

In this system, when fire or smoke is detected, the buzzer sounds the alarm. In addition, there is a tank for spraying water through a water pump. If there is a fire, the door will open automatically and sprinkle water. If the fire goes out, the door will automatically close.

4.2 Car Parking System

The car parking system is very useful for parking in the supermarket and university. In this system, LCD is used, so you can see how many cars are occupied. Therefore, it is possible to know whether the space is free/not free.

4.2.1 Block Diagram for Car Parking System

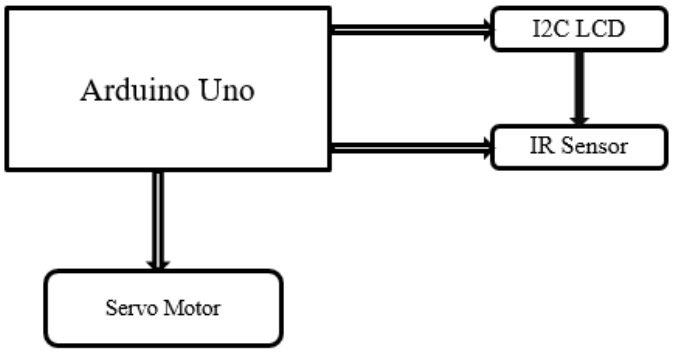


Figure 4.5: Block Diagram of Car Parking System

4.2.2 Circuit Diagram for Car Parking System

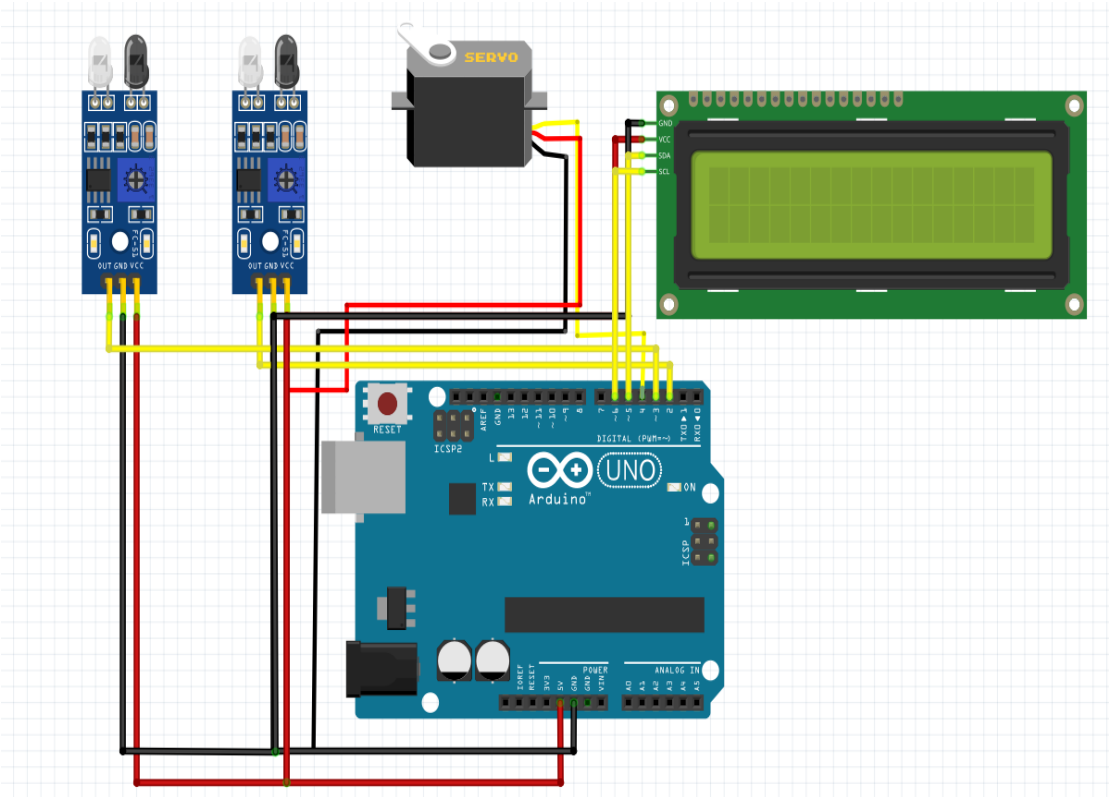


Figure 4.6: Circuit Diagram for Car Parking System

4.2.3 System Flow Diagram for Car Parking System

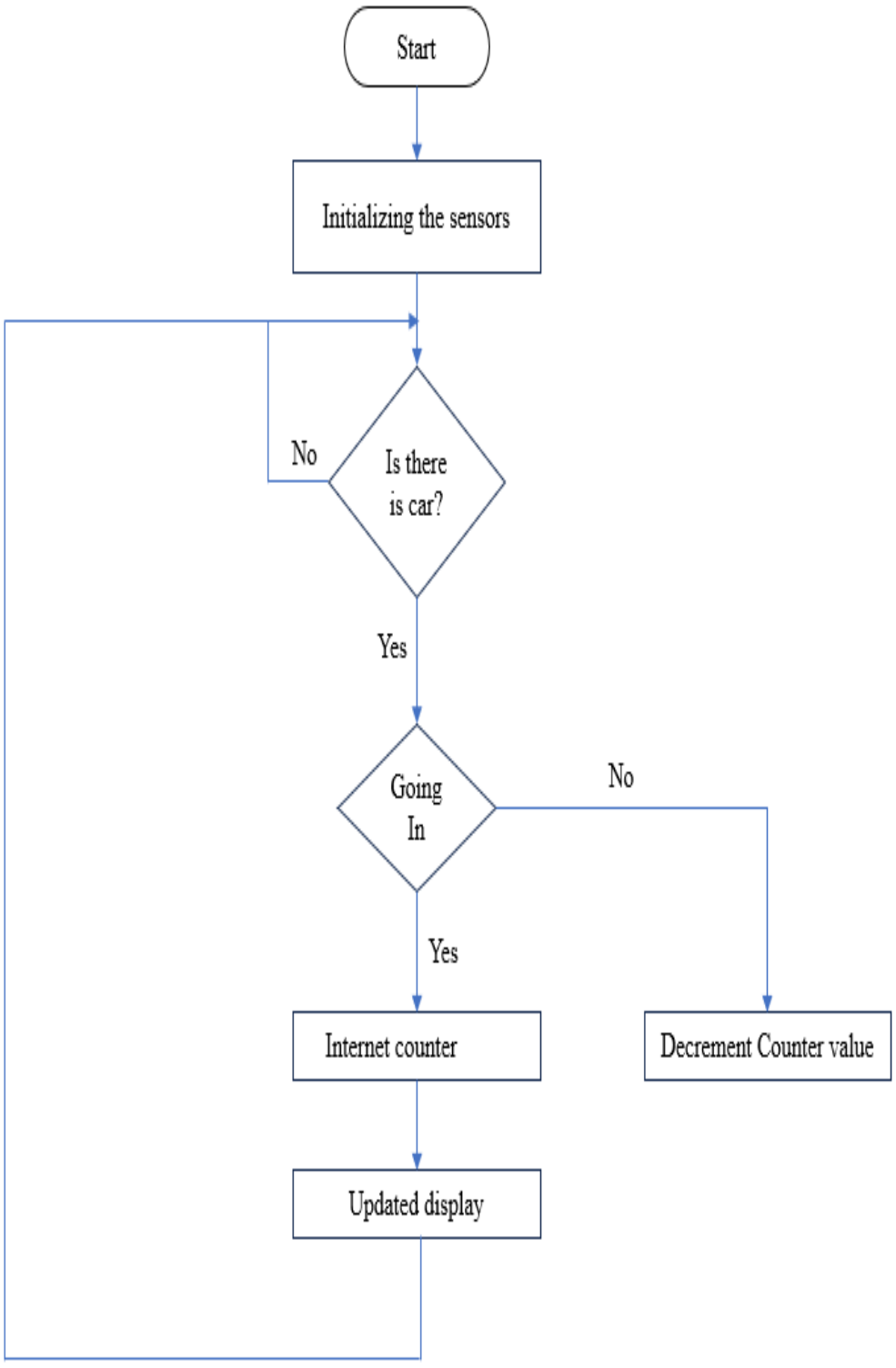


Figure 4.7: System Flow Diagram for Car Parking System

Table 4.2: Car Parking Pin Assignment Table

No	Device's Name	Device's Pins	Arduino Uno's Pin Number
1	IR Sensor (1) IR Sensor (2)	VCC Pin GND Pin OUT Pin	5V Pin GND Pin D2 Pin
2	Servo Motors	Control Signal VCC GND	D3 Pin 5V Pin GND Pin
3	I2C LCD-Display	GND VCC SDA SCL	GND Pin 5V Pin A4 Pin A5 Pin

4.2.4 Implementation for Car Parking System

In this system, there is an LCD to show whether there is an empty car parking space and how many cars are already occupied. Every time a car enters and exits, the number on the LCD display will change, and the servo motor will open and close the door automatically.

4.3 Machine Room System

This system is only for the room with computers. It is used to control the security of this room. There is no security system at the main door, so you can use this system to control security in the machine room and sever room where important data is stored.

4.3.1 Block Diagram for Machine Room System

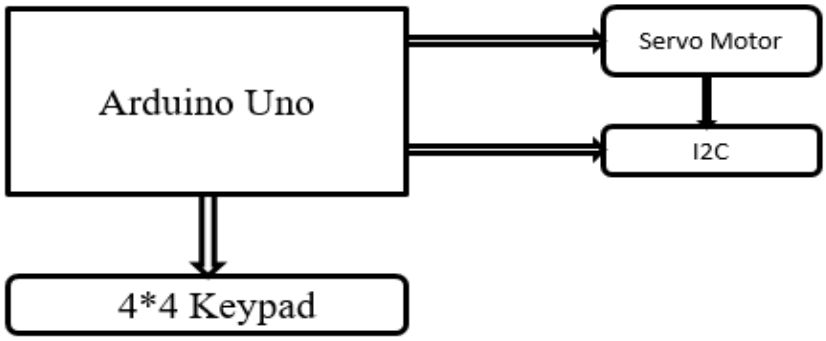


Figure 4.8: Block Diagram of Machine Room System

4.3.2 Circuit Diagram for Machine Room System

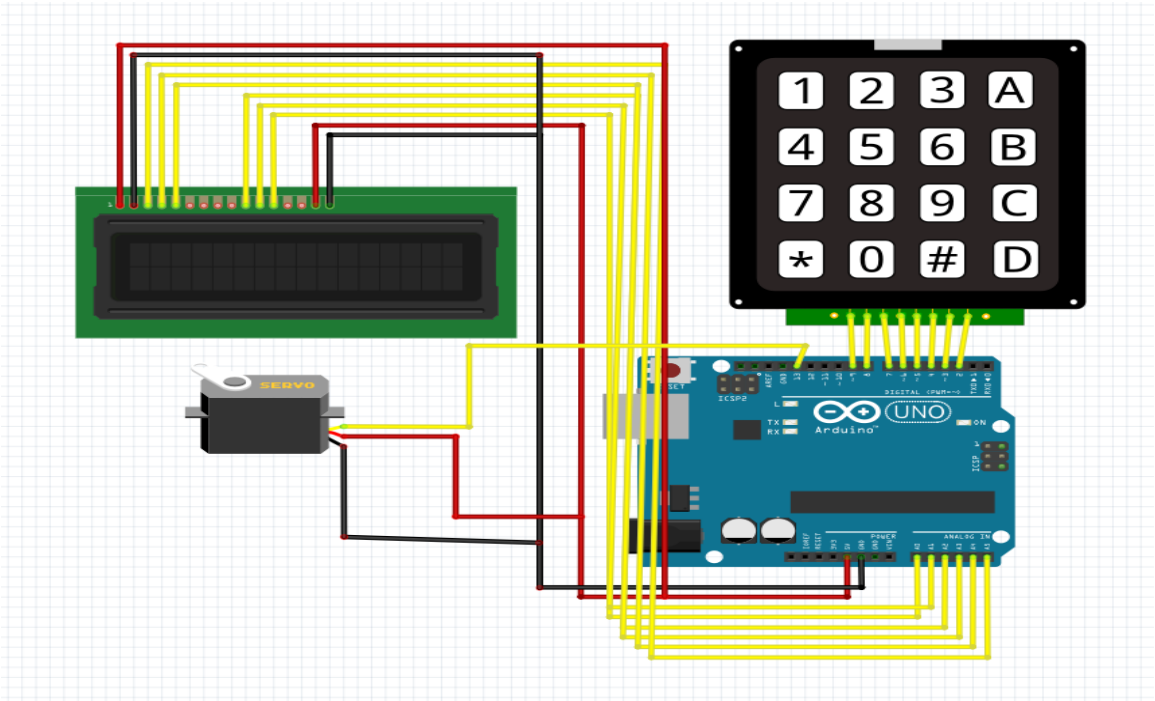


Figure 4.9: Circuit Diagram for Machine Room System

4.3.3 System Flow Diagram for Machine Room System

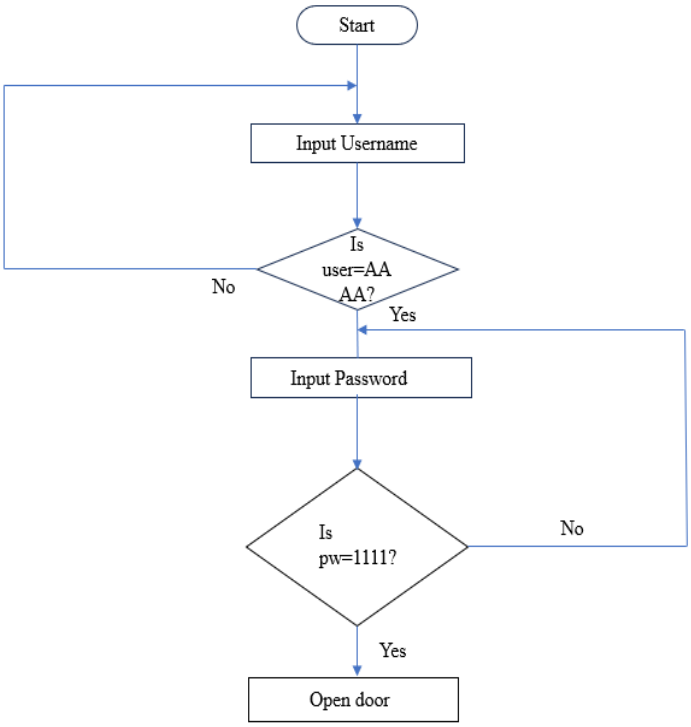


Figure 4.10: System Flow Diagram for Machine Room System

Table 4.3: Machine Room Pin Assignment Table

No	Device's Name	Device's Pins	Arduino Uno's Pin Number
1	LCD Display	VSS	GND Pin
		VCC	5V Pin
		VEE	GND Pin
		RS	Pin Not Used.
		RW	Pin Not Used.
		Enable	Pin Not Used.
		D0	A0
		D1	A1
		D2	A2
		D3	A3
		D4	A4
		D5	A5
		D6	Pin Not Used.
		D7	Pin Not Used.
		LED+	5V Pin
		LED-	GND Pin
2	Keypad	R1, R2, R3, R4	D9, D8, D7, D6
		C1, C2, C3, C4	D5, D4, D3, D2
3	Servo Motor	Control Signal	D13
		VCC	5V Pin
		GND	GND Pin

4.3.4 Implementation for Machine Room System

In this project, since there is no security control at the main door, this system is made because we want to maintain security in important rooms such as the server room and machine room. If you want to enter this room, you must enter the user id and password on the keypad. If these 2 facts are correct, the room door will open.

4.4 Main Door System

The door will open when the sensor detects the object. Some doors can be opened and closed manually. Using this door saves time, it relieves people's fatigue.

4.4.1 Block Diagram for Main Door System

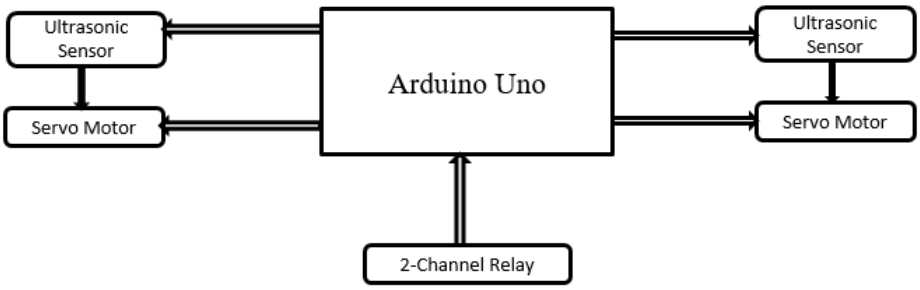


Figure 4.11: Block Diagram of Main Door System

4.4.2 Circuit Diagram for Main Door System

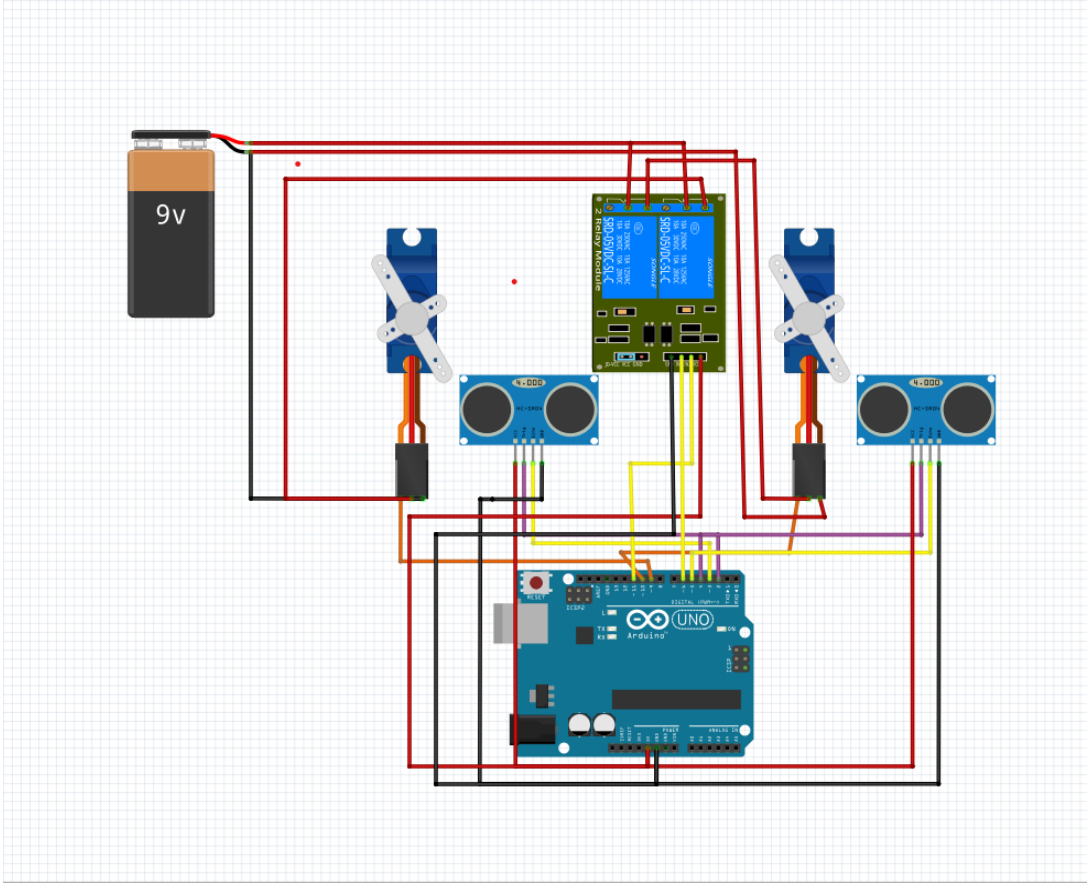


Figure 4.12: Circuit Diagram of Main Door System

4.4.3 System Flow Diagram for Main Door System

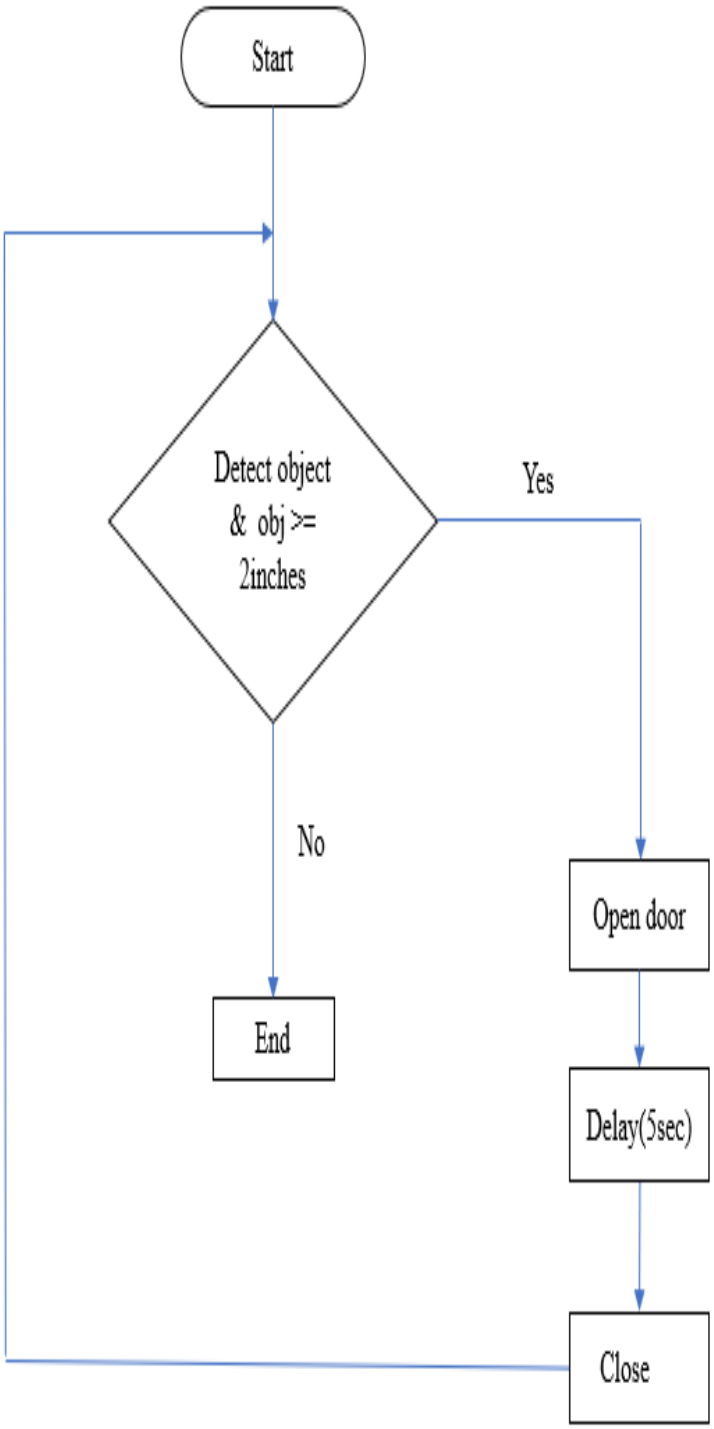


Figure 4.13: System Flow Diagram for Main Door System

Table 4.4: Machine Room Pin Assignment Table

No	Device's Name	Device's Pin	Arduino Uno's Pin Number
1	Ultrasonic Sensor (For Open)	Trig Pin Echo Pin VCC Pin & GND Pin	D2 D3
2	Ultrasonic Sensor (For Close)	Trig Pin Echo Pin VCC Pin & GND Pin	D4 D5
3	2-Channel Relay Module	IN-1 Pin IN-2 Pin VCC Pin & GND Pin Normally Open Pin Common Pin Normally Close Pin	D6 D11 Battery's GND Battery's VCC
4	Servo Motor (For Open)	Signal Pin VCC Pin & GND Pin	D9
5	Servo motor (For Close)	Signal Pin VCC Pin & GND Pin	D10
6	Battery	VCC Pin & GND Pin	
7	Push Button	Normally Open Terminal (2 Pins) Normally Close Terminal (2 Pins)	D12

4.4.4 Implementation for Main Door System

In our project, there are 2 doors for entry and exit. In the main door, the door will automatically open whenever it sees an object. After a short delay, the door will automatically close again.

4.5 Attendance System

This system has been made to know the attendance list of students. List data will be stored in the Google sheet and can be retrieved when you need it. Due to the cost of using this system, it saves time when calculating the attendance percentage.

4.5.1 Block Diagram for Attendance System

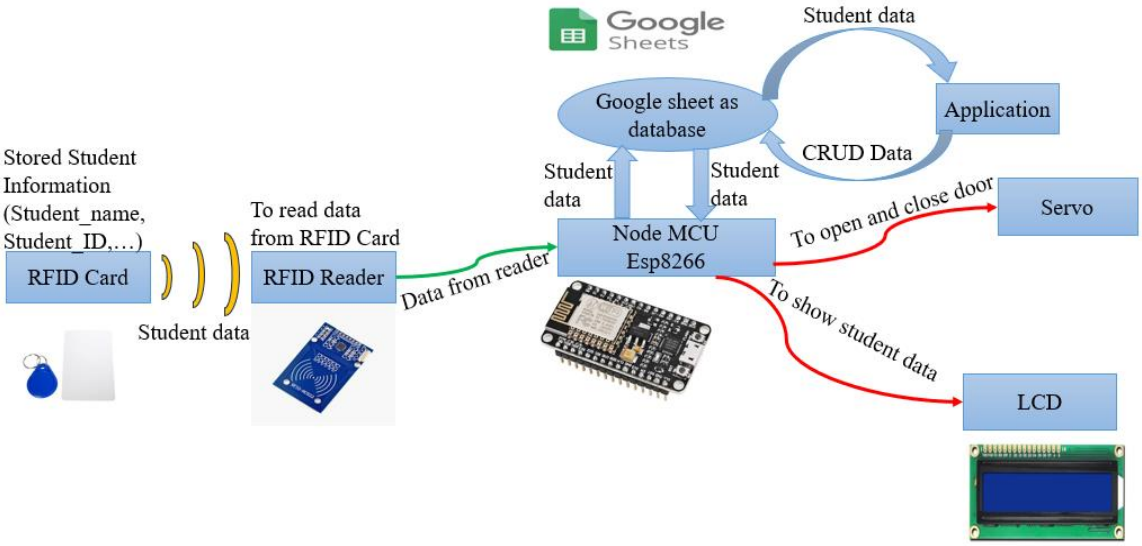


Figure 4.14: Block Diagram of Attendance System

4.5.2 Circuit Diagram for Attendance System

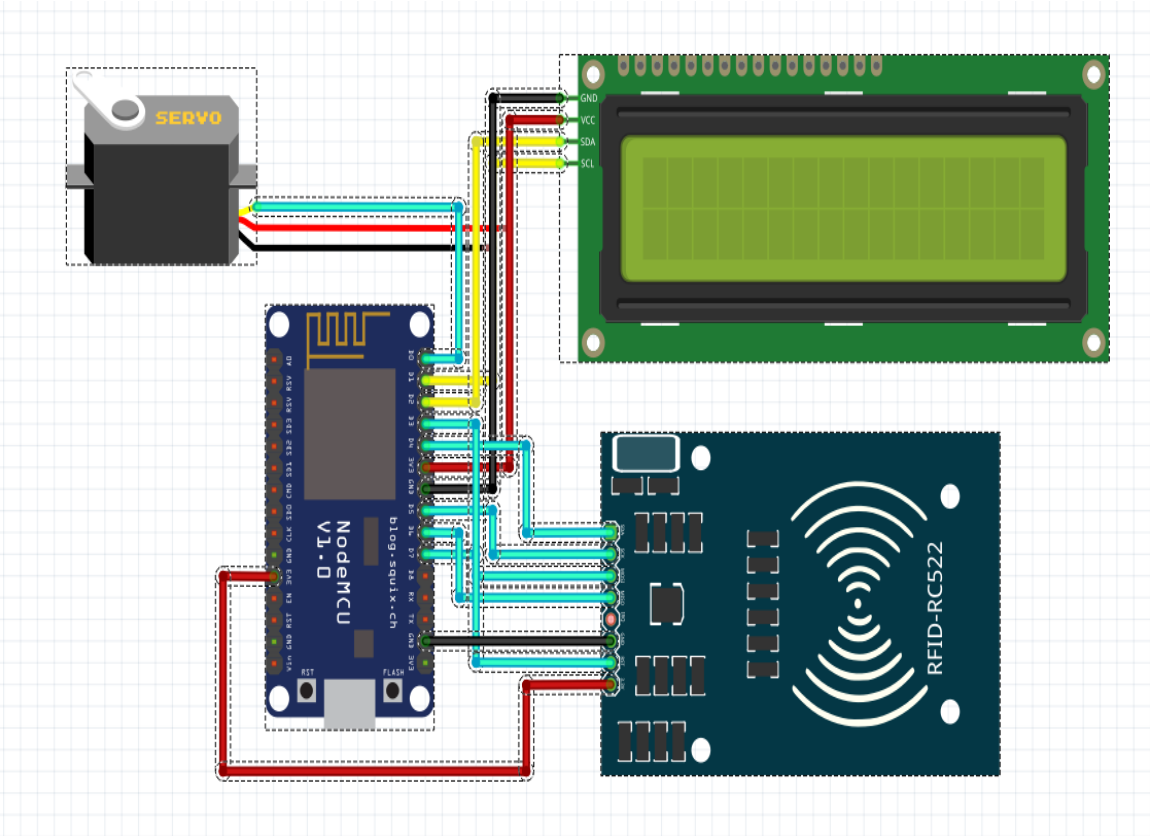


Figure 4.15: Circuit Diagram of Attendance System

4.5.3 System Flow Diagram for Attendance System

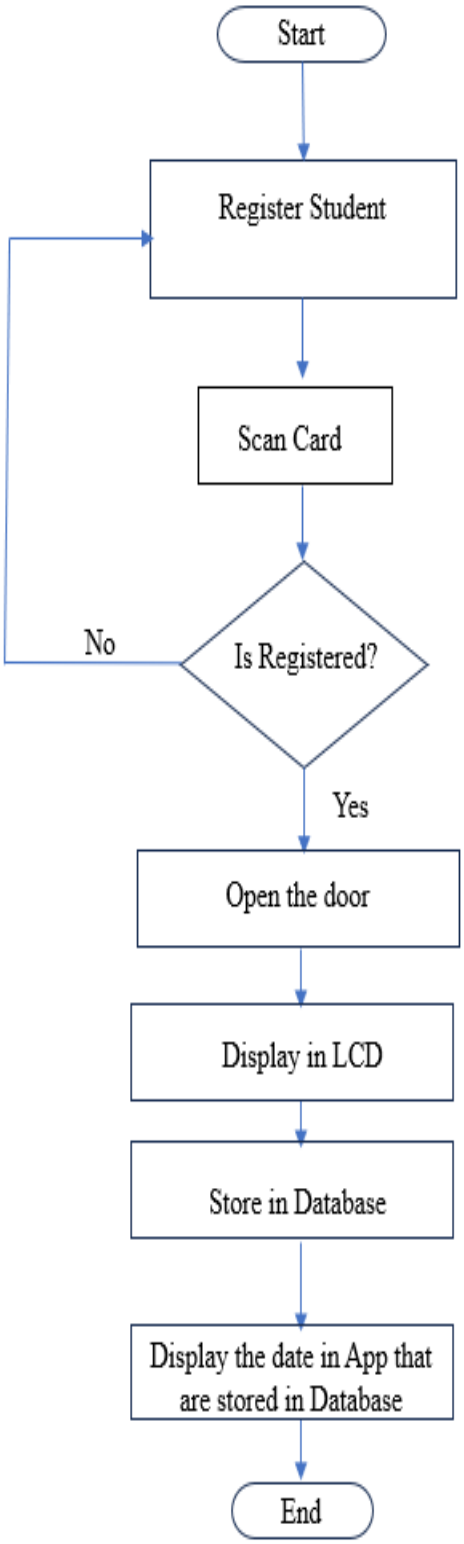


Figure 4.16: System Flow Diagram for Attendance System

Table 4.5: Attendance Pin Assignment Table

No	Device's Name	Device's Pins	Nose MCU's Pin Number
1	LCD Display	SDA Pin	D2
		SCL Pin	D1
		VCC Pin	VCC
		GND Pin	GND
2	RFID Card and Tags	SS Pin	D4
		SCK Pin	D5
		MOSI Pin	D7
		MIOS Pin	D6
		GND Pin	GND
		RST Pin	D3
		VCC Pin	VCC
3	Servo Motor	Signal Pin	D0
		VCC Pin	VCC
		GND Pin	GND

4.5.4 Implementation for Attendance System

Students' attendance is recorded in every university. Most of them are recorded manually. In this project, students' attendance can be recorded with RFID cards. If students want to enter the classroom or machine room, they must scan the RFID card and then enter. Student name, roll no, entry time, exit time will be linked to the database and stored in a google sheet.

4.6 IoT Control App System

Our IoT control app system can be used anywhere, anytime. It is self-developed with a new user interface using MIT Converter as an app that allows you to control the system with your own device by just having internet access at any time.

4.6.1 Block Diagram for IoT Control App System

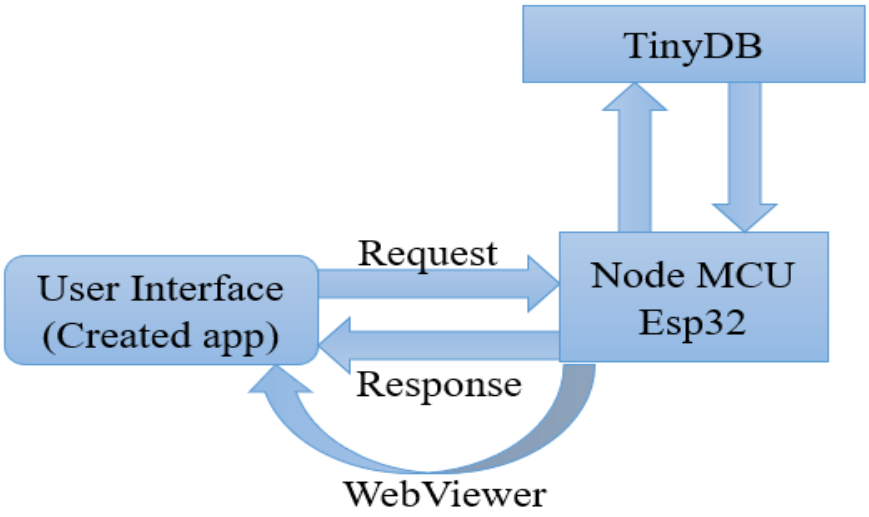


Figure 4.17: Block Diagram of IoT Control App System

4.6.2 Circuit Diagram for IoT Control App System

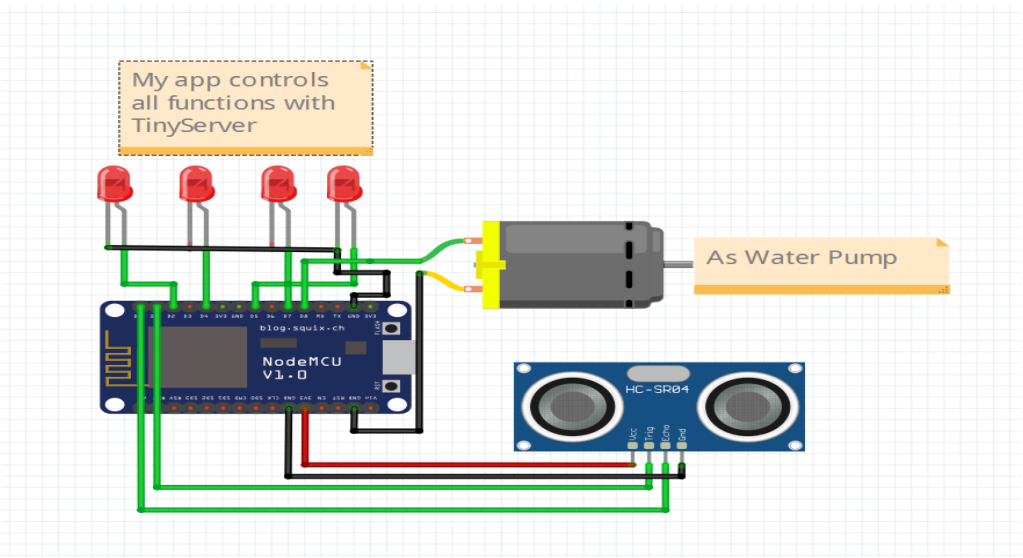


Figure 4.18: Circuit Diagram of IoT Control App System

4.6.3 System Flow Diagram for IoT Control App System

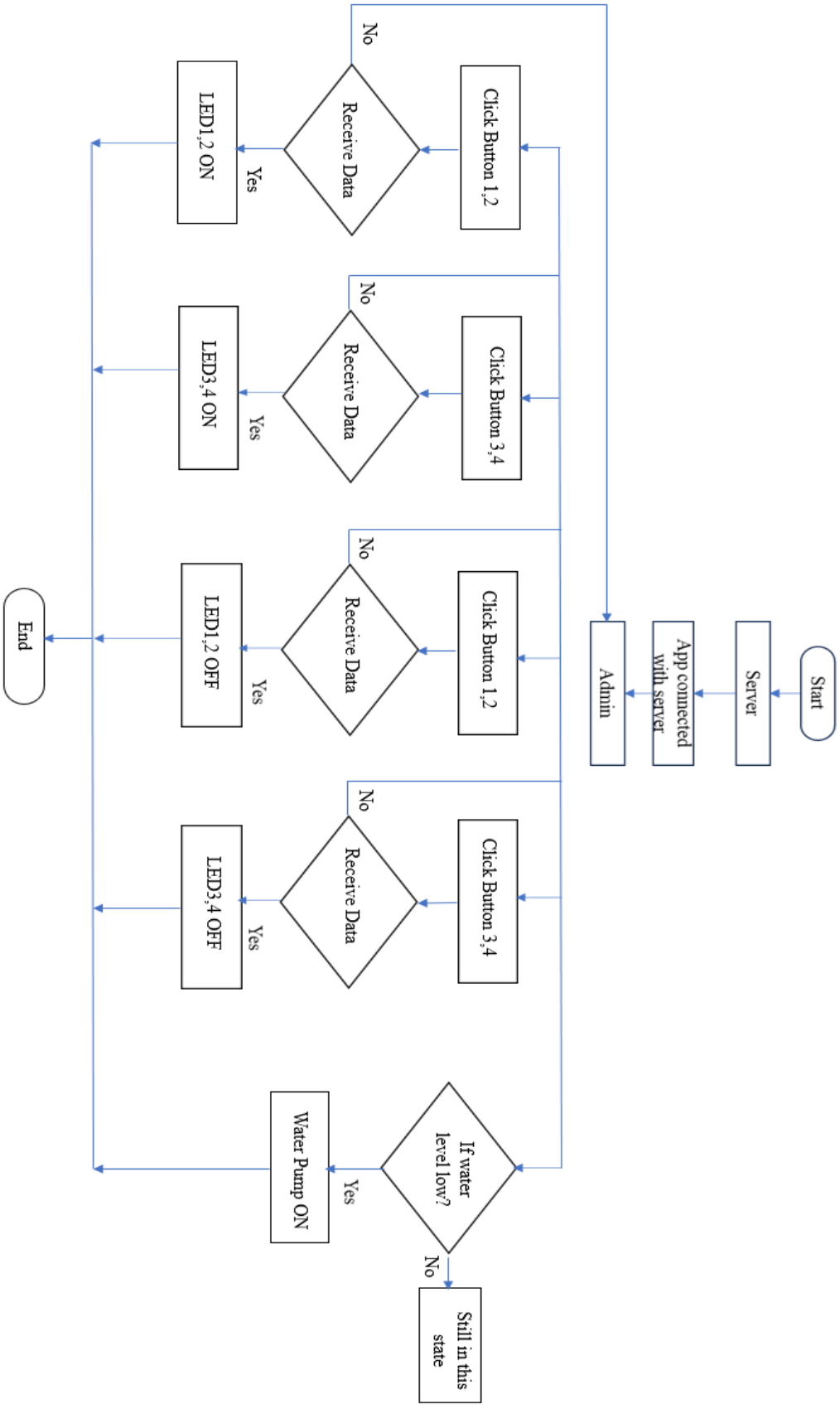


Figure 4.19: System Flow Diagram for IoT Control App System

Table 4.6: IoT Control App Pin Assignment Table

No	Device's Name	Device's Pins	Node MCU's Pin	Relay
1	LED 1	Anode Pin Cathode pin	D0 Pin GND	
2	LED 2	Anode Pin Cathode pin	D1 Pin GND	
3	LED 3	Anode Pin Cathode pin	D2 Pin GND	
4	LED 4	Anode Pin Cathode pin	D3 Pin GND	
5	Water Pump	VCC Pin GND Pin		Common Pin
6	Relay	IN 1 Pin VCC Pin GND Pin	D4 Pin VCC GND	
7	Battery	VCC Pin GND Pin		Normally Close Water pump's GND

4.6.4 Implementation for IoT Control App System

To turn on/off the lights in schools or homes, it is time consuming to search for the light switch and turn it on/off. For that, we have written an application. In this application, you can easily turn on and off the led by pressing a button. This application is also included to measure the water level in the water tanks and to add water if necessary.

CHAPTER 5

CONCLUSION

In conclusion, modern universities play a crucial role in shaping the future by providing access to higher education, fostering innovation, and conducting valuable research. They serve as hubs of knowledge, diversity, and collaboration, preparing students to tackle the challenges of the 21st century and contribute to the advancement of society, However, they also face ongoing challenges such as accessibility, affordability, and adapting to rapidly changing educational landscapes. The continued evolution of modern university will be essential to meet the needs of an ever-changing world.

5.1 Benefits and Limitations

- ⇒ Access to knowledge: Universities are hubs of knowledge, providing students with access to a wide range of academic resources, including libraries, and expert faculty members.
- ⇒ Diverse Learning Opportunities: Modern understanding offers a diverse range of programs and courses, allowing students to explore various fields of study.
- ⇒ Research Opportunities: Universities are often at the forefront of research and innovation.
- ⇒ Class Size and Personalized Attention: In large universities, class sizes can be substantial, making it harder for students to get personalized attention from professors. This can impact the quality of education for some.
- ⇒ Skills Gap and Job Market Relevance: Some argue that universities might not always adequately prepare students for the rapidly evolving job market, leading to a potential skills gap between academic knowledge and practical application.
- ⇒ It's important to note that while modern universities have these limitations, they also continuously strive to address these issues and provide the best possible education and resources for their students.

REFERENCES

⇒ https://microcontrollerslab.com/esp32-controller-android-mit-app-inventor/

⇒ https://www.ijraset.com/research-paper/rfid-based-attendance-system

⇒ https://projecthub.arduino.cc/info/in-and-out-automatic-door-sensor-546648

⇒ [1] Reuben, J.A., 1996. The making of the modern university: Intellectual transformation and the marginalization of morality. University of Chicago Press.

⇒ [2] Weisz, G., 2014. The emergence of modern universities in France, 1863-1914 (Vol. 522). Princeton University Press.

⇒ [3] Clark, B.R., 1995. Places of inquiry: Research and advanced education in modern universities. Univ of California Press.

⇒ [4] Bickel, R.D. and Lake, P.F., 1999. he Rights and Responsibilities of the Modern University: Who Assumes the Risks of College Life?. Education Review.

⇒ [5] Mansfield, E. and Lee, J.Y., 1996. The modern university: contributor to industrial innovation and recipient of industrial R&D support. Research policy, 25(7), pp.1047-1058.

APPENDIX A: Code for Main Door System (Arduino Uno)

```
#include <Servo.h>
const int openServoPin = 9;
const int closeServoPin = 10;
const int openUltrasonicTrigger = 2;
const int openUltrasonicEcho = 3;
const int closeUltrasonicTrigger = 4;
const int closeUltrasonicEcho = 5;
const int openRelayPin = 6;
const int closeRelayPin = 11;
const int presenceThreshold = 10; // Adjust this based on your needs

Servo openServo;
Servo closeServo;

void setup() {
  openServo.attach(openServoPin);
  closeServo.attach(closeServoPin);
  openServo.write(0); // Initialize open servo to closed position
  closeServo.write(0); // Initialize close servo to closed position

  pinMode(openUltrasonicTrigger, OUTPUT);
  pinMode(openUltrasonicEcho, INPUT);
  pinMode(closeUltrasonicTrigger, OUTPUT);
  pinMode(closeUltrasonicEcho, INPUT);

  pinMode(openRelayPin, OUTPUT);
  digitalWrite(openRelayPin,LOW);
  pinMode(closeRelayPin, OUTPUT);
  digitalWrite(closeRelayPin,LOW);
  pinMode(12,INPUT);
}

void loop() {
  // Check for presence near open door
  if (getUltrasonicDistance(openUltrasonicTrigger, openUltrasonicEcho) < pres-
enceThreshold || digitalRead(12)==HIGH) {
    openDoor();
  }
  if (getUltrasonicDistance (closeUltrasonicTrigger, closeUltrasonicEcho) < pres-
enceThreshold) {
    closeDoor ();}}
void openDoor () {
  digitalWrite(openRelayPin, HIGH); // Turn on the open relay
  openServo.write(130); // Set servo angle to open the door
  delay(8000); // Adjust delay as needed
  digitalWrite(openRelayPin, LOW);
  openServo.write(0); // Turn off the open relay
}
```

```
void closeDoor() {
  digitalWrite(closeRelayPin, HIGH); // Turn on the close relay
  closeServo.write(180); // Set servo angle to close the door
  delay(8000); // Adjust delay as needed
  digitalWrite(closeRelayPin, LOW);
  closeServo.write(0); // Turn off the close relay
}

long getUltrasonicDistance(int triggerPin, int echoPin) {
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  return pulseIn(echoPin, HIGH) / 58; // Convert time to distance in cm
}
```

APPENDIX B: Code for Car Parking System (Arduino Uno)

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);  //Change the HEX address
#include <Servo.h>

Servo myservo1;
int IR1 = 2;
int IR2 = 4;

int Slot = 4;
     //Enter Total number of parking Slots
int flag1 = 0;
int flag2 = 0;
void setup() {
  lcd.begin(16,2);
  lcd.backlight();
pinMode(IR1, INPUT);
pinMode(IR2, INPUT);

myservo1.attach(3);
myservo1.write(100);

lcd.setCursor (0,0);
lcd.print("     ARDUINO    ");
lcd.setCursor (0,1);
lcd.print(" PARKING SYSTEM ");
delay (2000);
lcd.clear();
}

void loop(){

if(digitalRead (IR1) == LOW && flag1==0){
if(Slot>0){flag1=1;
if(flag2==0){myservo1.write(0); Slot = Slot-1;}
}else{
lcd.setCursor (0,0);
lcd.print("    SORRY :(    ");
lcd.setCursor (0,1);
lcd.print("  Parking Full  ");
delay (3000);
lcd.clear();
}
}

if(digitalRead (IR2) == LOW && flag2==0){flag2=1;
if(flag1==0){myservo1.write(0); Slot = Slot+1;}
}
```

```
if(flag1==1 && flag2==1){
delay (1000);
myservo1.write(100);
flag1=0, flag2=0;
}

lcd.setCursor (0,0);
lcd.print("      WELCOME!      ");
lcd.setCursor (0,1);
lcd.print("   Slot Left: ");
lcd.print("[");
lcd.print(Slot);
lcd.print("]");

}
```

APPENDIX C: Code for Fire Alarm System (Arduino Uno)

```
#include <Servo.h>
const int flameSensorPin = 2;       // Flame sensor analog pin  // Relay control pin for
const int pumpRelayControlPin = 4;   // Relay control pin for water pump
const int buzzerPin = 9;
Servo servo;

void setup() {
  pinMode(flameSensorPin, INPUT);

  pinMode(pumpRelayControlPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  digitalWrite(buzzerPin,LOW);
  servo.attach(7);
  servo.write(0); // Initialize the servo to a closed position
   // Turn off the servo relay initially
    // Turn off the pump relay initially
  Serial.begin(9600);
}

void loop() {
  int flameValue = digitalRead(flameSensorPin);

  if (flameValue == LOW) {
    Serial.println("Flame detected! Activating response.");
    servo.write(180);

    // Open the door
    digitalWrite(buzzerPin, HIGH);
    digitalWrite(pumpRelayControlPin, HIGH);

    Serial.println("Fire response completed.");
  }
  else{
    digitalWrite(pumpRelayControlPin, LOW);
    delay(3000);
    servo.write(0);
    digitalWrite(buzzerPin, LOW);
  }

  delay(1000); // Wait for a second before checking again
}
```

APPENDIX D: Code for Machine Room System (Arduino Uno)

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

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);

Servo servo;

const char* correctPasscode = "1234";
const char* correctPasscode1 = "5678";
char enteredPasscode[5];

int door=9;
char userInput[5];
const char* user1 = "AAAA";
const char* user2 = "ABAA";

int userInputIndex = 0;
bool waitingForPasscode = false;

void setup() {
  lcd.begin(16, 2);
  lcd.print("Enter User Name:");
  pinMode(door,INPUT);
  digitalWrite(door,LOW);
  servo.attach(13);
  servo.write(0);
}
void clearEnteredPasscode() {
  memset(enteredPasscode, 0, sizeof(enteredPasscode));
  lcd.clear();
  lcd.print("Enter User Name:");}
void clearUserInput() {
  memset(userInput, 0, sizeof(userInput));
  userInputIndex = 0;
  lcd.setCursor(0, 1);
lcd.print("    "); // Clear input line
```

```
}
void openDoor() {
  lcd.clear();
  lcd.print("Access granted");
  servo.write(90);
  delay(6000);
  servo.write(0);
  delay(1000);
  clearUserInput();
  waitingForPasscode = false;
  clearEnteredPasscode();
}

void accessDenied() {
  lcd.clear();
  lcd.print("Access denied");
  delay(1000);
  clearUserInput();
  waitingForPasscode = false;
  clearEnteredPasscode(); // Clear the user input sequence as well
}
void loop() {
 char key = keypad.getKey();
  if (key != NO_KEY) {
    if (!waitingForPasscode) {
      if (key == 'A' || key == 'B' || key == 'D') {
        userInput[userInputIndex] = key;
        userInputIndex++;
        lcd.setCursor(userInputIndex - 1, 1);
        lcd.print(key);

        if (userInputIndex == 4 && strcmp(userInput, user1) == 0){
          lcd.clear();
          lcd.print("Enter Passcode:");
          clearUserInput();
          waitingForPasscode = true;
        }
      } else {
        accessDenied();
      }
    } else {
      if (key == '#') {
        if (userInputIndex == 4) {
          if (strcmp(userInput, correctPasscode) == 0)
{
```

```
      openDoor();
    }
    else {
      accessDenied();
    }
  } else {
    accessDenied();
  }
}
else if (key == 'C') {
  clearUserInput();
}
else if (userInputIndex < 4) {
  lcd.setCursor("Enter Passcode:" + userInputIndex, 1);
  lcd.setCursor(userInputIndex, 1);
  lcd.print('*');
  userInput[userInputIndex] = key;
  userInputIndex++;
}}}}
```

APPENDIX E

APPENDIX E1: Code for Attendance System in extension of AppScript in GoogleSheet (Node MCU)

```
// Enter Spreadsheet ID here
var SS = SpreadsheetApp.openById('1V36HL_sTXFl3IRgry_jzU-5LqaoAH-wzlhcRJnqyOu8I');
var timezone = "Asia/Yangon";
var hours = 0;
var str = "";

function doPost(e) {

  var parsedData;
  var result = {};

  try {
    parsedData = JSON.parse(e.postData.contents);
  }
  catch(f){
    return ContentService.createTextOutput("Error in parsing request body: " + f.message);
  }

  if (parsedData !== undefined){
    var flag = parsedData.format;
    if (flag === undefined){
      flag = 0;
    }

    var sheet = SS.getSheetByName(parsedData.sheet_name); // sheet name to publish data to is specified in Arduino code
    var dataArr = parsedData.values.split(","); // creates an array of the values to publish

    var Curr_Date = Utilities.formatDate(new Date(), timezone, "MM/dd/yyyy"); // gets the current date
    var Curr_Time = Utilities.formatDate(new Date(), timezone, "hh:mm:ss a"); // gets the current time
    //var Curr_Date = new Date(new Date().setHours(new Date().getHours() + hours));
    //var Curr_Time = Utilities.formatDate(Curr_Date, timezone, 'HH:mm:ss');
    // comming from Arduino code
    var value0 = dataArr [0];  //Student ID
    var value1 = dataArr [1];  //First Name
    var value2 = dataArr [2];  //Last Name
    var value3 = dataArr [3];  //Phone Number
    var value4 = dataArr [4];  //Address
    var value5 = dataArr [5];  //Gate Number
```

```
  //-------------------------------------------------------------------------------------------------
  /* STEP1 - This piece of code searches for the student ID in the attendance sheet. If the student ID is found,
  it gets the row number of that student ID and retrieves their time-out data.
  */
  var data = sheet.getDataRange().getValues();
  var row_number = 0;
  var time_out = "";
  //for(var i = data.length - 1; i >= 0; i--){  // Search last occurrence
  for(var i = 0; i < data.length ; i++){  // Search first occurrence of student id
   if(data[i][0] == value0){ //data[i][0] i.e. [0]=Column A, Student_id
     row_number = i+1;
     time_out = data[i][2] //time out [2]=Column C

     console.log("row number: "+row_number); //print row number
     console.log("time out: "+time_out); //print row number
  break; //go outside the loop
   }
  }
  /* STEP2 - Next, it checks if the time-out variable is empty. If it is empty, the current time is added to the
  time-out field and a message is returned to NodeMcu.
  */
  if(row_number > 0){
   if(time_out === ""){
     sheet.getRange("C"+row_number).setValue(Curr_Time);
     str = "Success"; // string to return back to Arduino serial console
     return ContentService.createTextOutput(str);
   }
  }
  //Otherwise,the attendance is recorded as usual using the code written below
  //-------------------------------------------------------------------------------------------------
     switch (parsedData.command) {
   case "insert_row":

     sheet.insertRows(2); // insert full row directly below header text

     sheet.getRange('A2').setValue(value0);     // publish STUDENT ID to cell A2
     sheet.getRange('B2').setValue(Curr_Time);  // publish TIME IN to cell B2
     //sheet.getRange('C2').setValue();         // publish TIME OUT to cell C2
     sheet.getRange('D2').setValue(value5);     // publish GATE NUMBER to cell D2
     sheet.getRange('E2').setValue(Curr_Date);  // publish DATE to cell E2
     sheet.getRange('F2').setValue(value1);     // publish FIRST NAME cell F2
     sheet.getRange('G2').setValue(value2);     // publish LAST NAME cell G2
     sheet.getRange('H2').setValue(value3);     // publish PHONE NUMBER cell H2
     sheet.getRange('I2').setValue(value4);     // publish ADDRESS cell I2
```

```
        str = "Success"; // string to return back to Arduino serial console
        SpreadsheetApp.flush();
        break;

      case "append_row":

        var publish_array = new Array(); // create a new array

        publish_array [0] = value0;    // publish Student ID to cell A2
        publish_array [1] = Curr_Time; // publish Time In to cell B2
        publish_array [3] = Curr_Date; // publish current date to cell D2
        publish_array [4] = value1;    // publish First Name cell E2
        publish_array [5] = value2;    // publish Last Name cell F2

        sheet.appendRow(publish_array); // publish data in publish_array after the last
row of data in the sheet

        str = "Success"; // string to return back to Arduino serial console
        SpreadsheetApp.flush();
        break;
    }

    return ContentService.createTextOutput(str);
  } // endif (parsedData !== undefined)

  else {
    return ContentService.createTextOutput("Error! Request body empty or in incorrect
format.");
  }
}
```

APPENDIX E2: Code for Attendance System for Student Registration (Node MCU)

```
#include <SPI.h>
#include <MFRC522.h>
//-------------------------------------------------
//GPIO 0 --> D3
//GPIO 2 --> D4
const uint8_t RST_PIN = D3;
const uint8_t SS_PIN = D4;
//-------------------------------------------------
MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
int blockNum = 4;
byte bufferLen = 18;
byte readBlockData[18];
MFRC522::StatusCode status;

void setup() {
  //-------------------------------------------------------
  //Initialize serial communications with PC
  Serial.begin(9600);
  //-------------------------------------------------------
  //Initialize SPI bus
  SPI.begin();
  //-------------------------------------------------------
  //Initialize MFRC522 Module
  mfrc522.PCD_Init();
  Serial.println("Scan a MIFARE 1K Tag to write data...");
  //-------------------------------------------------------
}


void loop() {

  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
  //-------------------------------------------------------------------------------
  /* Look for new cards */
  /* Reset the loop if no new card is present on RC522 Reader */
  if (!mfrc522.PICC_IsNewCardPresent()) { return; }
  //-------------------------------------------------------------------------------
  if (!mfrc522.PICC_ReadCardSerial()) { return; }
  //-------------------------------------------------------------------------------
  Serial.print("\n");
  Serial.println("**Card Detected**");
 Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
```

```
  Serial.print("\n");
  /* Print type of card (for example, MIFARE 1K) */
  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));

  //-------------------------------------------------------------------------
  byte buffer[18];
  byte len;
  //wait until 20 seconds for input from serial
  Serial.setTimeout(20000L);
  //MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
  Serial.println(F("---------------------------------------"));
  Serial.println(F("Enter Student ID, ending with #"));
  len = Serial.readBytesUntil('#', (char *)buffer, 16);
  //add empty spaces to the remaining bytes of buffer
  for (byte i = len; i < 16; i++) buffer[i] = ' ';
  blockNum = 4;
  WriteDataToBlock(blockNum, buffer);
  ReadDataFromBlock(blockNum, readBlockData);
  dumpSerial(blockNum, readBlockData);
  Serial.println(F("---------------------------------------"));
  Serial.println(F("Enter Name, ending with #"));
  len = Serial.readBytesUntil('#', (char *)buffer, 16);
  for (byte i = len; i < 16; i++) buffer[i] = ' ';
  blockNum = 5;
  WriteDataToBlock(blockNum, buffer);
  ReadDataFromBlock(blockNum, readBlockData);
  dumpSerial(blockNum, readBlockData);
  Serial.println(F("---------------------------------------"));
  Serial.println(F("Enter Year, ending with #"));
  len = Serial.readBytesUntil('#', (char *)buffer, 16);
  for (byte i = len; i < 16; i++) buffer[i] = ' ';
 blockNum = 6;
  WriteDataToBlock(blockNum, buffer);
  ReadDataFromBlock(blockNum, readBlockData);
  dumpSerial(blockNum, readBlockData);
  //MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
  Serial.println(F("---------------------------------------"));
  Serial.println(F("Enter Phone Number, ending with #"));
  len = Serial.readBytesUntil('#', (char *)buffer, 16);
  for (byte i = len; i < 16; i++) buffer[i] = ' ';
blockNum = 8;
  WriteDataToBlock(blockNum, buffer);
  ReadDataFromBlock(blockNum, readBlockData);
  dumpSerial(blockNum, readBlockData);
  Serial.println(F("---------------------------------------"));
  Serial.println(F("Enter Address, ending with #"));
  len = Serial.readBytesUntil('#', (char *)buffer, 16);
```

```
  for (byte i = len; i < 16; i++) buffer[i] = ' ';
  blockNum = 9;
  WriteDataToBlock(blockNum, buffer);
  ReadDataFromBlock(blockNum, readBlockData);
  dumpSerial(blockNum, readBlockData);
}

void WriteDataToBlock(int blockNum, byte blockData[]) {
  //Serial.print("Writing data on block ");
  //Serial.println(blockNum);
  //-----------------------------------------------------------------------------
  /* Authenticating the desired data block for write access using Key A */
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Authentication failed for Write: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  //-----------------------------------------------------------------------------
  else {
    //Serial.print("Authentication OK - ");
  }
  status = mfrc522.MIFARE_Write(blockNum, blockData, 16);
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Writing to Block failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  } else {
    //Serial.println("Write OK");
  }
  //-----------------------------------------------------------------------------
}
void ReadDataFromBlock(int blockNum, byte readBlockData[]) {
  //Serial.print("Reading data from block ");
  //Serial.println(blockNum);
  //-----------------------------------------------------------------------------
  /* Prepare the ksy for authentication */
  /* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory */

for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
  //-----------------------------------------------------------------------------
  /* Authenticating the desired data block for Read access using Key A */
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));
  //-----------------------------------------------------------------------------
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Authentication failed for Read: ");
```

```
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  } else {
    //Serial.print("Authentication OK - ");
  }
  //-------------------------------------------------------------------------
  /* Reading data from the Block */
  status = mfrc522.MIFARE_Read(blockNum, readBlockData, &bufferLen);
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Reading failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  } else {
    //readBlockData[16] = ' ';
    //readBlockData[17] = ' ';
    //Serial.println("Read OK");
  }
  //-------------------------------------------------------------------------
void dumpSerial(int blockNum, byte blockData[]) {
  Serial.print("\n");
  Serial.print("Data saved on block");
  Serial.print(blockNum);
  Serial.print(": ");
  for (int j = 0; j < 16; j++) {
    Serial.write(readBlockData[j]);
  }
  Serial.print("\n");

  //Empty readBlockData array
  for (int i = 0; i < sizeof(readBlockData); ++i)
    readBlockData[i] = (char)0;  //empty space
}
```

APPENDIX E3: Code for Attendance System for Student Attendance (Node MCU)

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <MFRC522.h>
#include <HTTPSRedirect.h>
#include <Servo.h>

#include<Wire.h>
#include<LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
//----------------------------------------------------------------------------------------------------
---
// Enter Google Script Deployment ID:
const char *GScriptId = "AKfycbyVLc6ZQgSJfWALVer-
haz3G4xmYkWC44lx6eoFamewb9QkNcCbVtAB1rRPUL6f1hQI";
String gate_number = "Present";
Servo myservo;  // Create a Servo object

//----------------------------------------------------------------------------------------------------
---
// Enter network credentials:
const char* ssid     = "RectorOffice";
const char* password = "*123*RectorOffice#";
//----------------------------------------------------------------------------------------------------
---
// Enter command (insert_row or append_row) and your Google Sheets sheet name
(default is Sheet1):
String payload_base =  "{\"command\": \"insert_row\", \"sheet_name\": \"Sheet1\",
\"values\": ";
String payload = "";
//----------------------------------------------------------------------------------------------------
---
// Google Sheets setup (do not edit)
const char* host       = "script.google.com";
const int   httpsPort   = 443;
const char* fingerprint = "";
String url = String("/macros/s/") + GScriptId + "/exec";
HTTPSRedirect* client = nullptr;
//------------------------------------------------------------
// Declare variables that will be published to Google Sheets
String student_id;
//------------------------------------------------------------
int blocks[] = {4,5,6,8,9};
#define total_blocks  (sizeof(blocks) / sizeof(blocks[0]))
```

```
#define RST_PIN  0  //D3
#define SS_PIN   2  //D4
#define BUZZER   4  //D2
//------------------------------------------------------------
MFRC522 mfrc522(SS_PIN, RST_PIN);
MFRC522::MIFARE_Key key;
MFRC522::StatusCode status;
int blockNum = 2;
byte bufferLen = 18;
byte readBlockData[18];
//------------------------------------------------------------

void setup() {
  //------------------------------------------------------------
  Serial.begin(9600);
  delay(10);
  myservo.attach(D0);
  myservo.write(0);
  Serial.println('\n');
  //------------------------------------------------------------
  SPI.begin();
  //------------------------------------------------------------
  //initialize lcd screen
  lcd.init();
  // turn on the backlight
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0); //col=0 row=0
  lcd.print("Connecting to");
  lcd.setCursor(0,1); //col=0 row=0
  lcd.print("WiFi...");
  //------------------------------------------------------------
  // Connect to WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to ");
  Serial.print(ssid); Serial.println(" ...");

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println('\n');
  Serial.println("WiFi Connected!");
  //Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  // Use HTTPSRedirect class to create a new TLS connection
  client = new HTTPSRedirect(httpsPort);
  client->setInsecure();
  client->setPrintResponseBody(true);
  client->setContentTypeHeader("application/json");
```

```
  lcd.clear();
  lcd.setCursor(0,0); //col=0 row=0
  lcd.print("Connecting to");
  lcd.setCursor(0,1); //col=0 row=0
  lcd.print("Google ");
  delay(5000);
  Serial.print("Connecting to ");
  Serial.println(host);
  bool flag = false;
  for(int i=0; i<5; i++){
    int retval = client->connect(host, httpsPort);
    //*************************************************
    if (retval == 1){
      flag = true;
      String msg = "Connected. OK";
      Serial.println(msg);
      lcd.clear();
      lcd.setCursor(0,0); //col=0 row=0
      lcd.print(msg);
      delay(2000);
      break;
    }
    //*************************************************
    else
      Serial.println("Connection failed. Retrying...");
    //*************************************************
  }
  //-------------------------------------------------------
  if (!flag){
    //_____________________________________________________
    lcd.clear();
    lcd.setCursor(0,0); //col=0 row=0
    lcd.print("Connection fail");
    //_____________________________________________________
    Serial.print("Could not connect to server: ");
    Serial.println(host);
    delay(5000);
    return;
  }
  delete client;    // delete HTTPSRedirect object
  client = nullptr; // delete HTTPSRedirect object
}
  static bool flag = false;
if (!flag){
    client = new HTTPSRedirect(httpsPort);
    client->setInsecure();
    flag = true;
    client->setPrintResponseBody(true);
    client->setContentTypeHeader("application/json");
  }
```

```
  if (client != nullptr){
    if (!client->connected()){
      int retval = client->connect(host, httpsPort);
      if (retval != 1){
        Serial.println("Disconnected. Retrying...");
        lcd.clear();
        lcd.setCursor(0,0); //col=0 row=0
        lcd.print("Disconnected.");
        lcd.setCursor(0,1); //col=0 row=0
        lcd.print("Retrying...");
        return; //Reset the loop
      }
    }

  }
  else{Serial.println("Error creating client object!"); Serial.println("else");}
  //-----------------------------------------------------------------
  lcd.clear();
  lcd.setCursor(0,0); //col=0 row=0
  lcd.print("Scan your Tag");

  mfrc522.PCD_Init();
  if ( ! mfrc522.PICC_IsNewCardPresent()) {return;}
  /* Select one of the cards */
  if ( ! mfrc522.PICC_ReadCardSerial()) {return;}
  /* Read data from the same block */
  Serial.println();
  Serial.println(F("Reading last data from RFID..."));
  //-----------------------------------------------------------------
  String values = "", data;
  for (byte i = 0; i < total_blocks; i++) {
    ReadDataFromBlock(blocks[i], readBlockData);
    //*************************************************

if (i == 0){
      data = String((char*)readBlockData);
      data.trim();
      student_id = data;
      values = "\"" + data + ",";
    }
    /*else if(i == total_blocks-1){
data.trim();
      values += data + "\"}";
    }*/
    //*************************************************
    else{
      data = String((char*)readBlockData);
      data.trim();
      values += data + ",";
    }
```

```
  }
  values += gate_number + "\"}";
  payload = payload_base + values;
  lcd.clear();
  lcd.setCursor(0,0); //col=0 row=0
  lcd.print("Publishing Data");
  lcd.setCursor(0,1); //col=0 row=0
  lcd.print("Please Wait...");
  //---------------------------------------------------------------
  // Publish data to Google Sheets
  Serial.println("Publishing data...");
  Serial.println(payload);
  if(client->POST(url, host, payload)){
    // do stuff here if publish was successful
    Serial.println("[OK] Data published.");
    lcd.clear();
    lcd.setCursor(0,0); //col=0 row=0
    lcd.print("Student ID:"+student_id);
    lcd.setCursor(0,1); //col=0 row=0
    lcd.print("Present");
    myservo.write(180);     // Move servo to 180 degrees
    delay(3000);
    myservo.write(0);     // Move servo to 180 degrees

  }
  //---------------------------------------------------------------
  else{
    // do stuff here if publish was not successful
    Serial.println("Error while connecting");
    lcd.clear();
    lcd.setCursor(0,0); //col=0 row=0
    lcd.print("Failed.");
    lcd.setCursor(0,1); //col=0 row=0
    lcd.print("Try Again");
  }
  //---------------------------------------------------------------
  // a delay of several seconds is required before publishing again
  Serial.println("[TEST] delay(5000)");
  delay(5000);
}
void ReadDataFromBlock(int blockNum, byte readBlockData[])
{
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }

  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK){
     Serial.print("Authentication failed for Read: ");
```

```
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else {
    Serial.println("Authentication success");
  }
  status = mfrc522.MIFARE_Read(blockNum, readBlockData, &bufferLen);
  if (status != MFRC522::STATUS_OK) {
    Serial.print("Reading failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else {
    readBlockData[16] = ' ';
    readBlockData[17] = ' ';
    Serial.println("Block was read successfully");
  }
}
```

APPENDIX F: Code for IoT Control App System (Node MCU)

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>

MDNSResponder mdns;

const char* ssid = "RectorOffice";
const char* password = "*123*RectorOffice#";

WebServer server(80);
String webpage = "";

int trigger_pin = 18;
int echo_pin   = 19;

int led1 = 4;
int led2 = 5;
int led3 = 2;
int led4 = 22;
int pump = 21;

String page = "";
int distance_cm;

void setup(void) {

 server.on("/", []() {
    webpage = "<head><meta http-equiv=\"refresh\" content=\"3\"></head><cen-
ter><h3>Current water level: " + String(distance_cm) + "</h3></center>";
    server.send(200, "text/html", webpage);
  });
  pinMode(trigger_pin, OUTPUT);
  pinMode(echo_pin, INPUT);
  delay(1000);
  pinMode(led1, OUTPUT);
  digitalWrite(led1, LOW);
  pinMode(led2, OUTPUT);
  digitalWrite(led2, LOW);
  pinMode(led3, OUTPUT);
  digitalWrite(led3, LOW);
  pinMode(pump, OUTPUT);
  digitalWrite(pump, LOW);
  pinMode(led4, OUTPUT);
  digitalWrite(led4, LOW);
  Serial.begin(115200);
 WiFi.begin(ssid, password);
  Serial.println("");
  while (WiFi.status() != WL_CONNECTED)
```

```
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
   if (mdns.begin("esp")) {
    Serial.println("MDNS responder started");
  }
  server.on("/", []() {
    server.send(200, "text/html", webpage);
  });

  server.on("/led1ON", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);

  });

  server.on("/led1OFF", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);

  });

  server.on("/led2ON", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led3, HIGH);

  });
  server.on("/led2OFF", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led3, LOW);

  });

  server.on("/PumpON", []() {

  server.send(200, "text/html", webpage);
  digitalWrite(pump, HIGH);
  });
server.on("/PumpOFF", []() {
  server.send(200, "text/html", webpage);
  digitalWrite(pump, LOW);
  });
```

```
  server.on("/Led3ON", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led4, HIGH);
  });

  server.on("/Led3OFF", []() {
    server.send(200, "text/html", webpage);
    digitalWrite(led4, LOW);
  });
server.begin();
  Serial.println("HTTP server started");
}
void loop(void) {
  server.handleClient();
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger_pin, LOW);
  long duration = pulseIn(echo_pin, HIGH);
  distance_cm = (duration / 2) / 29.09;
  Serial.println(distance_cm);
  delay(1000);
}
```