Zagazig University

Faculty of Engineering

Electronics and Communications Engineering Department

Smart Campus

Presented by:

Ibrahim saber Mohamed

Ahmed Nasser Ahmed

Ahmed Mahmoud Mohamed

Taqwa Hamed Goda

Kholoud Khaled Atia

Radwa Refat Galal

Yasmeen Mohamed Saad

Supervised by:

Prof. Fathy Farag

Dr. Abdelhamied Ashraf

2022

Table of Contents

List of Figures

Acknowledgment

We would like to express our full gratitude and own a deep sense of gratitude to our Graduation Project Supervisors **DR. Abdelhamied Ashraf** and **DR. Fathy Farag** for the advice, knowledge, insightful discussions, and suggestions throughout the entire journey, and for patiently guiding us through his processes. You made us believe that we have what it takes and the ability to reach even greater heights in our graduation project.

Abstract

Many organizations, including but not limited to colleges, schools, companies, and city halls, are looking forward to smart cities. For the past few years, the concept of the Internet of Things (IoT) has been a recurrent view of the technological environment where nearly every object is expected to be connected to the network. This infrastructure will progressively allow one to monitor and efficiently manage the environment. Until recent years, IoT applications have been constrained by limited computational capacity and efficient communications. Still, the emergence of new communication technologies allows us to overcome most of these issues.

This situation paves the way for the fulfillment of the Smart-City concept, where the cities become a fully efficient, monitored, and managed environment able to sustain the increasing needs of its citizens and achieve environmental goals and challenges. In this project, we deploy the concept of smart cities to build a reliable smart campus system. The developed work deploys the IoT paradigm to provide a fully automated smart campus. All designed parts are presented in this report and the obtained results are introduced and discussed.

Keywords: Smart city, Internet of Things, Fifth-generation cellular, smart metering.

1. Chapter 1: Introduction

1.1: Problem Statement

In traditional learning environment, higher education institutions compete on various levels to attract students. our lives are already embedded with smart technologies in our cars, stores, banks, and our homes. We interact with smart environments regularly. Yet, many universities are still inundated with siloed infrastructures that often do not serve or communicate outside of their associated stakeholders.

Modern students are digital natives, they want to have a well-rounded and seamless experience with technology that integrates into their everyday life and just does everything more smartly. So, it must be changed so that the old traditional campus becomes a modern smart campus that talks to students. A smart campus is an interface between smart homes and smart cities. Through intelligently connected everyday university life, like many other smart apps and experiences, it's about communicating with other systems, devices, or the Internet.

To improve efficiencies and enhance the experience of students and staff on campus, smart campus has been introduced. It offers entirely new opportunities to the organization itself with a potential increase in operational efficiency, cost savings, and the improvement of public safety.

To this end, we consider developing a reliable smart campus-based on recent novel technologies to meet the recent requirements. The considered smart campus can be viewed as sub-systems that are introduced as follows.

Smart parking system: A smart parking system allows the user quick access which helps in the reduction of time in searching the parking spot, reducing traffic congestion. It can be used to monitor parking systems and exhibit the parking lot situation at any given moment.

Smart lighting system: It is introduced to increase the accuracy and efficiency of street lighting, and therefore solve the problem of accidents occurring in the nighttime is either caused by driver errors or malfunctioning streetlights The malfunctions can be caused by fused bulbs or damaged wiring. This project reduces this error hence reducing accidents significantly.

Smart metering system: It records information about the consumption of electric energy. then communicate the information to the consumer for greater clarity of consumption behavior, and electricity suppliers for system monitoring and customer billing.

Auto attendance system: It uses the face recognition algorithm for recording the attendance of students in a classroom. The idea of the project is very helpful so the teacher or doctor to have an excel sheet with the student attending the lecture.

Mask detection system: It detects whether people wear a mask or not. This is to control the infection in cases of pandemics.

1.2: Motivation

Our Motivation is to help increase the quality of life for all people. by using networked technology to improve cooperation, resource efficiency, security, and cost savings, and render the institution more integrated and pleasurable. Institutions may use a smart campus to integrate systems like smart lighting that control light intensity and save power, Smart parking to manage the cars and improve security by saving the driver's ID to help to trace him., Smart metering that controls power consumption, Smart attendance that record the present students using a camera, mask detection that permits only the student who wears a mask to enter the gate.

1.3: Aim and objectives

1.3.1: Aim

We aim to This project aims to focus on a smart campus as a basic model. There will be a centralized server that automatically performs tasks and take the decision to open the gate, control street lighting, provide the available parking spots on a map, and record class attendance. It can help to reduce energy consumption, reduce electricity costs and enhance security and resource management.

1.3.2: Objectives

- Being able to record information about the consumption of electric energy, voltage levels, Current, and power factors.
- Being able to reduce power consumption and cost .and able to easily control the devices that consumed large power
- Being able to control the intensity of the lighting by LDR.
- Being able to control street lighting using a real-time clock by setting the sunrise and sunset time.
- Being able to allow the user quick access which helps in the reduction of time in searching the parking spot, reducing traffic congestion.
- Being able to add security as every user has its id and save the time of entrance and existence in a cloud server.
- Being able to use the Face recognition algorithm for recording the attendance of students in a classroom.
- The ability for the teacher or doctor to have an excel sheet with the student attending the lecture.
- Being Able to detect people wearing a mask to reduce harmful effects on health and reduce the spreading of diseases.

Chapter 2: Smart Metering System

2.1: Objective

- The main objective of this system is to record information about the consumption of electric energy, voltage levels, Current, and power factors.
- communicate the information to the consumer for greater clarity of consumption behavior, and electricity suppliers for system monitoring and customer billing

2.2: System design and analysis

2.2.1: Top-level design

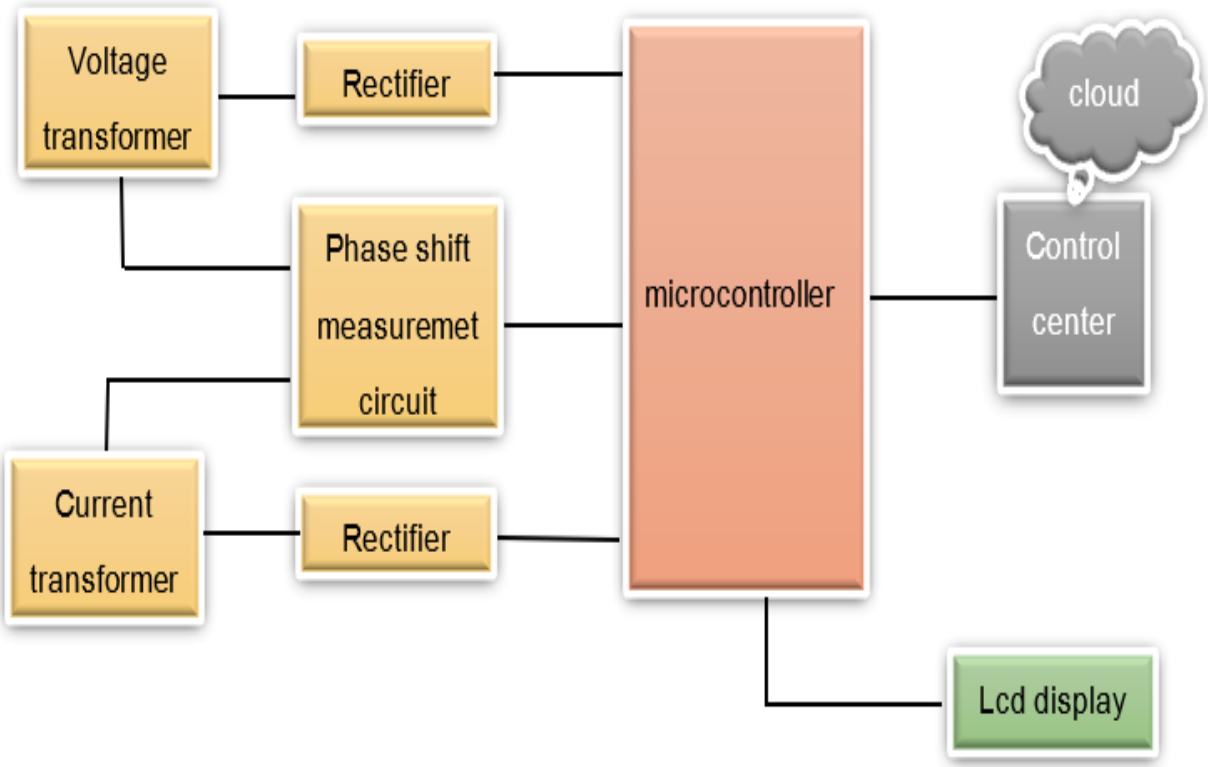


Figure 0-1 The System Architecture

2.2.2: Flow Chart

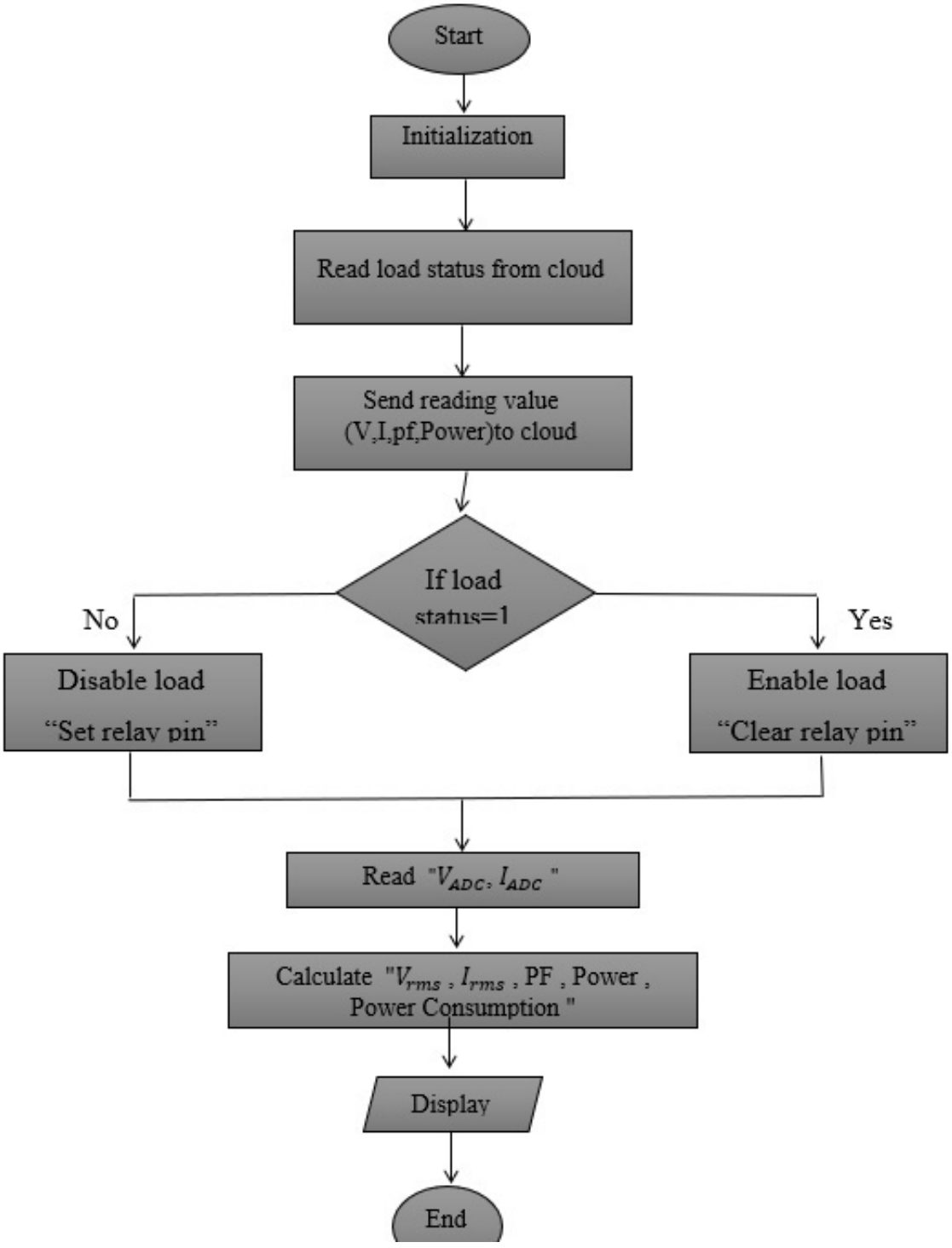


Figure 0-2 Flow Chart

2.2.3: Sequence

2.2.3.1: Sequence Diagram

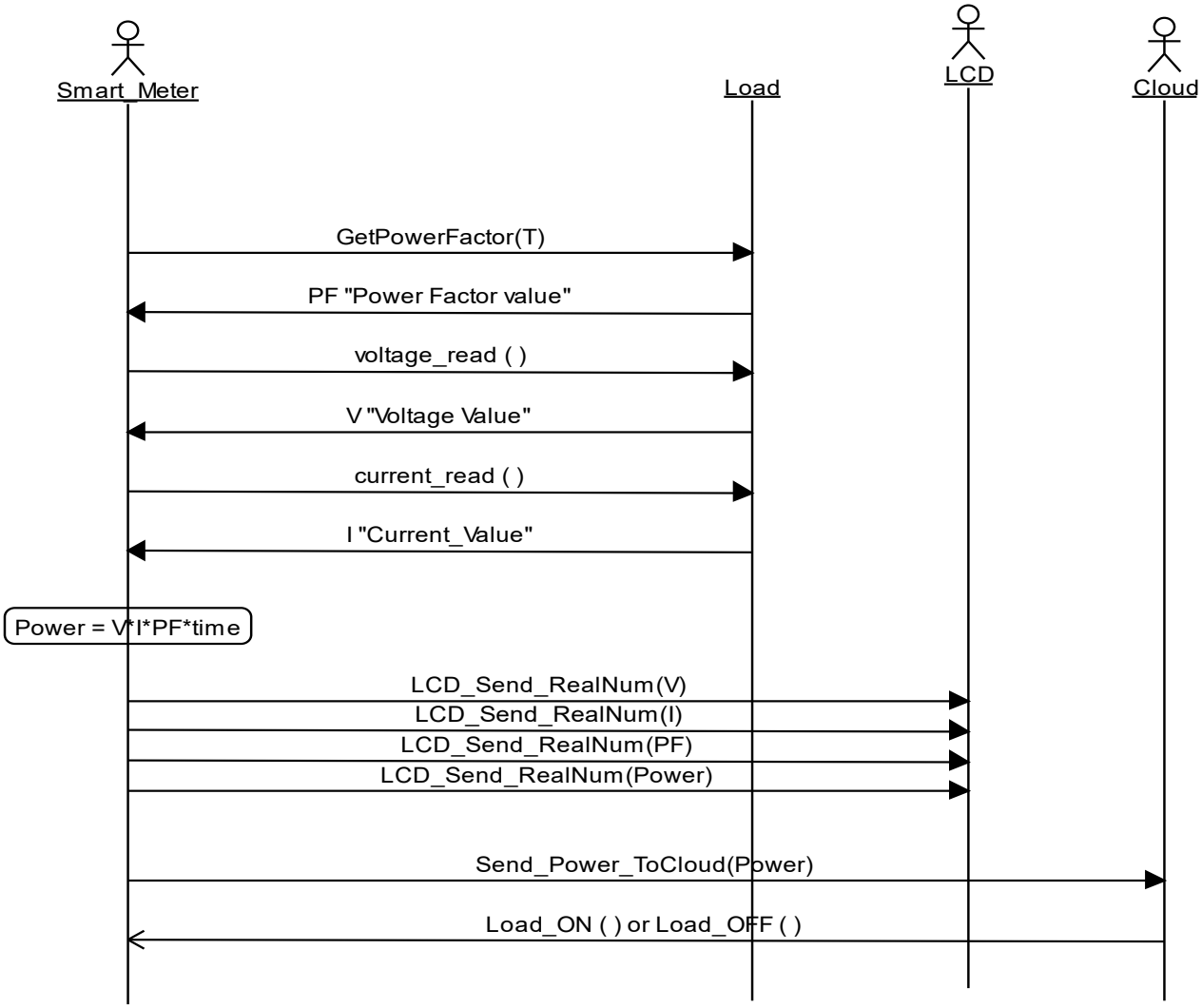


Figure 0-3 Sequence Diagram

2.2.3.2: Code Sequence

Begin

//initialize DIO

PA0 INPUT // for voltmeter

PA1 INPUT //for ammeter

PA2 INPUT // power factor 1

```
PA3 INPUT //power factor 2
PD0 OUTPUT // relay

Initialize  ADC
Initialize  LCD
Initialize  USART
Begin connection to cloud

        WHILE (TRUE)
                Read load status from cloud
                Send Values of Power, Voltage, and Current to Cloud
                IF (load_status)
                        Clear relay pin // enable Load
                ELSE
                        Set relay pin // disable Load
                ENDIF
                Read Voltage From load
                Read Current From load
                Read Power Factor From load
                Calculate Power in KWH (V*I*pf*measurement_time/3,600,000)
                Display Voltage on LCD
                Display Current on LCD
                Display Power Factor on LCD
                Display Power consumed on LCD
                Clear screen
        END While
END
```

2.2.4: Hardware

2.2.4.1: ADC

- According to ADC on ATMega32. The ATmega32 features a **10-bit** successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).
- The supplied CLK from the MCU which is 1MHZ can be prescaled from ADCSRA reg in our case we used a 1/16 Prescaler.

  ADC Frequency = F_cpu / Prescaler

  = 1Mhz / 16 = 62.5KHZ

  Conversion Time = 1/ADC Frequency

  = 1/62500 = 16 Microsecond.

- 13 ADC Cycles Required for One Conversion

  13 *16 Microsecond = **230 us** [Approximately]

- For these configs to calculate single-ended conversion, the result is **Resolution = 5V/ 1024 = 4.88 Mv**

- **ADC =Vin / Res = Vin * (1024 / 5V)** which is the content in the ADCL & ADCH regs

- **Conversion theory of ADC:**

  The conversion goes through 3 main steps

  - Sampling
  - Quantization
  - Coding

  This conversion is done by the Successive approximation ADC that comes built in most of the microcontrollers as it is suitable for

  - low-cost
  - medium to high-resolution applications
  - low power consumption

- **Successive Approximation ADC:**

  This ADC consists of:

  - A Sample and Hold circuit to acquire the input voltage, V_{in}.
  - An analog voltage comparator that compares V_{in} to the output of the internal DAC and outputs the result of the comparison to the successive-approximation register (SAR).
  - A successive approximation register subcircuit is designed to supply an approximate digital code of V_{in} to the internal DAC.
  - An internal reference DAC that, for comparison with V_{ref}, supplies the Comparator with an analog voltage equal to the digital code output of the SAR_{in}.

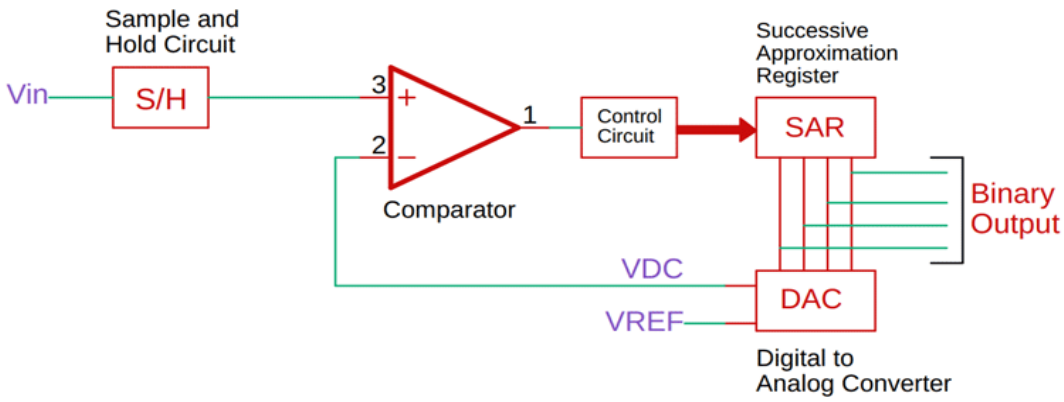


Figure 0-4 Successive Approximation ADC Circuit

- **working principle:**

  - The successive approximation register is initialized with (MSB) equal to a digital 1.
  - This code is fed into the DAC, which then supplies the analog equivalent of this digital code ($V_{ref}/2$) into the comparator circuit for comparison with the sampled input voltage.
  - If this analog voltage exceeds V_{in}, then the comparator causes the SAR to reset this bit; otherwise, the bit is left as 1.
  - Then the next bit is set to 1 and the same test is done, continuing this binary search until every bit in the SAR has been tested.
  - The resulting code is the digital approximation_of the sampled input voltage and is finally output by the SAR at the end of the conversion.

2.2.4.2: Timer0

- According to Timer/counter 0 on ATMega32.Timer/counter 0 is a general-purpose, single-channel counter 8_ bit timer/counter module 10-bit clock Prescaler

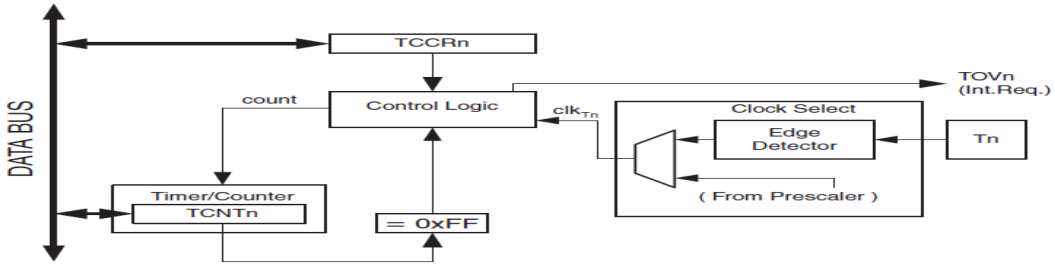


Figure 0-5 8-bit Timer/Counter Block Diagram

- At every counter, there is at least one compare register. When the counter value equals the value of the compare register, a specific action can be triggered. Actions can also be triggered when the timer value reaches an overflow.
- In an 8-bit timer, the register is 8 bits long (TCNT0) and thus can store a number from 0 to 255. This register has the property of increasing or decreasing its value without any intervention by the CPU at a rate frequently defined by the user. This frequency (at which the timer registers increases/decreases) is known as the Timer Frequency.
- A timer can be clocked by an internal or an external clock source selected in the control register (TCCR0). Set the timer frequency by “Prescaler” It is the method of generating the clock frequency for the TIMER from the CPU clock by dividing it by a fair number.
- The counter direction is always incrementing, and no counter clear is performed. Counter overruns when it passes its max (255) and then restarts from the bottom (0) the timer overflow flag (TOV0) will be set in the same timer clock cycle as the TCNT0 becomes zero. The TOV0 Flag in this case behaves like a ninth bit but is only set not cleared. Interrupt that automatically clearTOV0 flag.

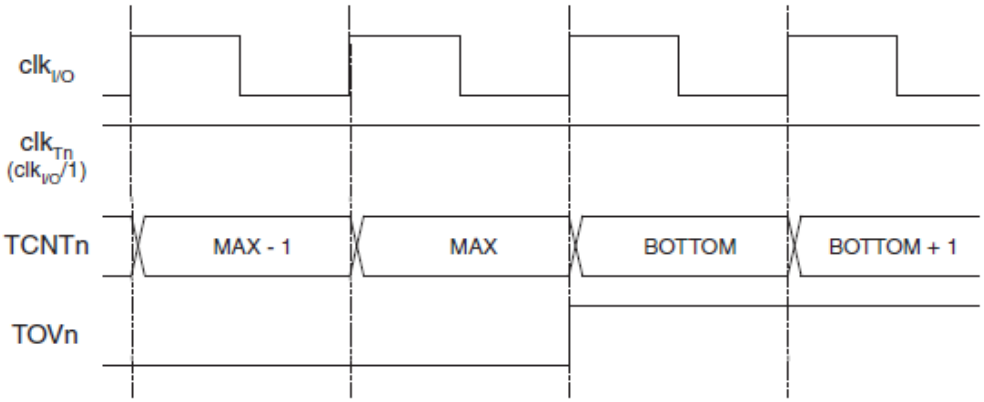


Figure 0-6 Timer/Counter Timing Diagram, no Pre-scaling

.2.4.3: Interrupt

- According to interrupt on atmega32, The AVR 8-bits microcontroller provides both internal and external interrupt sources.
- The internal interrupts are associated with the microcontroller's peripherals ex Timer/Counter, Analog-Digital Converter (ADC), USART, etc.
- The external interrupts are triggered via external pins. for an AVR 8-bit microcontroller. On this microcontroller there are four (4) external interrupts:
  - ✓ RESET interrupts: Triggered from pin 9.
  - ✓ External Interrupt 0 (INT0): Triggered from pin 16(PD2).
  - ✓ External Interrupt 1 (INT1): Triggered from pin 17(PD3).
  - ✓ External Interrupt 2 (INT2): Triggered from pin 3(PB2).

- **Important registers for external interrupts implementation:**

  - The SREG register enabled global interrupt is bit7, the I-bit, write a 1 to the specific interrupt bit to enable.

  - The GICR register contains the interrupt enable bits for all the external

  interrupts. Write a 1 to the specific interrupt bit to enable that interrupt.

  - The MCUCR registry special for interrupt 0,1 is used to make interrupts with (low level, high level, falling edge, rising edge).

  - The MCUCR registry special for interrupt 2 is used to make interrupts with (falling edge, rising edge).

  - The GIFR registry Set flag bit when the interrupt is enabled and clear when the ISR is executed.

- **Steps to Execute an Interrupt:**

  1- The microcontroller completes the execution of the current assembly instruction, clears the I -bit, and stores the address of the next instruction that should have been executed, general-purpose registers, and processor status register.

  2- The interrupt vector of the triggered interrupt is then loaded in the PC from the interrupt vector table and the microcontroller starts execution from that point up until reaches the end of the ISR.

  3- The address that was stored on the stack in step 1 is reloaded in the PC register. The status register and the general-purpose registers are popped from the stack and the I-bit is re-enabled.

  4- The micro-controller then continues executing the program.

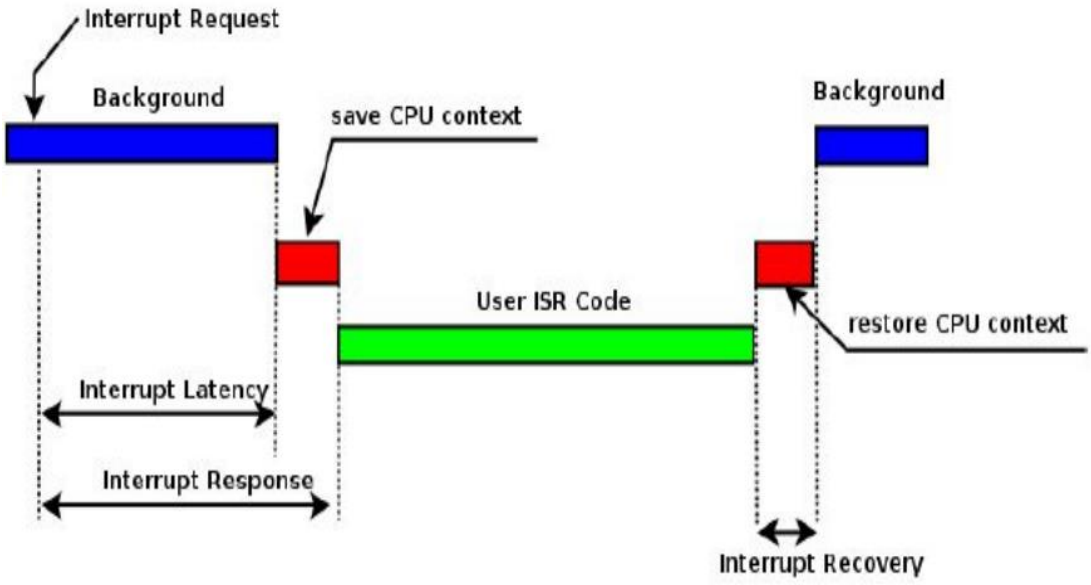


Figure 0-7 Interrupt Service Routine

2.2.4.6: Potential Transformer

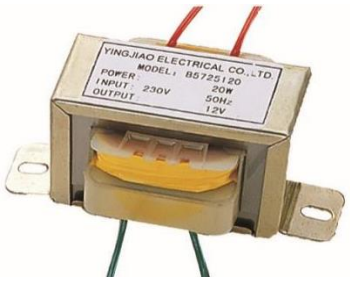


Figure 0-8 Potential Transformer

- It is a device that converts the energy of one form to another.
- VTs are used to monitor alternating AC or DC by measuring voltage directly or through a VT.
- VT is a parallel-connected type of instrument transformer
- It is used for stepping down the system voltage to a safe value which can be fed to low ratings meters and relays.
- The core of the transformer works to direct the path of the magnetic field between the primary and secondary coils to prevent wasted energy. Once the magnetic field reaches the secondary coil, it forces the electrons within it to move, creating an electric current via electromotive force (EMF).

2.2.4.7: Diode

Figure 0-9 Diode

- Diodes can be made of either of the two semiconductor materials, silicon, and germanium.
- Used diode to protect circuits by limiting the voltage and to also transform AC into DC.
- Also used are diodes as rectifiers, signal limiters, voltage regulators, switches, signal modulators, signal mixers, signal demodulators, and oscillators.
- The fundamental property of a diode is its tendency to conduct electric current in only one direction.
- Characteristics Of Diode:
    - ✓ Forward-biased Diode when there is a small drop of voltage across the diode.
    - ✓ Reverse-biased Diode when the battery's voltage is dropped completely.
    - ✓ Zero-biased Diode, when the voltage potential across the diode is zero.

2.2.4.8: Capacitor

Figure 0-10 Capacitor

- Uses Capacitors for storing electrical energy.
- One of the capacitor's most important features is the ability to resist voltage changes if the voltage applied to a capacitor changes suddenly.
- the capacitors smooth out the rectified utility AC, providing pure, battery-like DC.

2.2.4.9: Resistance

Figure 0-11 Resistance

- Resistance helps in manipulating the current that is flowing through the circuit.
- It helps in saving electric appliances from the excess current it leads to the breakdown of a circuit.
- It enables the measurement of current and voltage flowing through a circuit.

.2.4.10: Zener

Figure 0-12 Zener.

- It is a special type of diode designed to reliably allow current to flow "backward" when a certain set reverse voltage is reached.
- Zener is used as a Shunt voltage regulator for regulating voltage across small loads. The Zener diode is connected parallel to the load to make it reverse bias.
- And used to generate low-power stabilized supply rails from a higher voltage and provide reference voltages for circuits and protect circuits from overvoltage.

2.2.4.11: ACS712 Current sensor

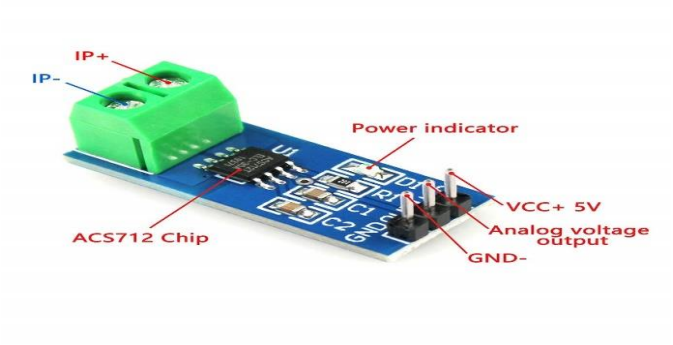


Figure 0-13 ACS712 Current sensor

- uses ACS712 IC to measure the current using the Hall Effect principle.
- Can measure both AC & DC.
- **Here's how the ACS712 work (Simplified):**

  Current flows through the onboard hall sensor circuit in its IC. The hall effect sensor detects the incoming current through its magnetic field generation. Once detected, the hall effect sensor generates a voltage proportional to its magnetic field that's then used to measure the amount of current.

  If volt= 2.5v indication no magnetic field ,no current

  If volt >2.5 indications to mine current

  If volt >2.5 indications of positive current

2.2.4.12: LM358

Figure 2-14 LM358

- It consists of two internally frequency compensated, high gain, independent op-amps.
- This IC is designed specially to operate from a single power supply over a wide range of voltages.
- The LM358 IC is available in a chip-sized package and applications of this op-amp include conventional op-amp circuits, DC gain blocks, and transducer amplifiers.
- LM358 IC is a good and standard operational amplifier.
- It can handle 3-32V DC supply & source up to 20mA per channel. This op-amp is apt if you want to operate two separate op-amps for a single power supply. It's available in an 8-pin DIP package.
- Pin Configuration of LM358 IC
  - ✓ Pin-1 and pin-8 are o/p of the comparator
  - ✓ Pin-2 and pin-6 are inverting i/ps
  - ✓ Pin-3 and pin-5 are non-inverting i/ps
  - ✓ Pin-4 is the GND terminal
  - ✓ Pin-8 is VCC+

2.2.4.13: - 16×2 LCD

Figure 2-15 16×2 LCD

- It is a type of flat panel display which uses liquid crystals in its primary form of operation. mostly depend on multi-segment LEDs.
- Used to display information.

2.3: Description

2.3.1: Measure Voltage

1-Step down the AC voltage:

- using potential transformer convert volt from 220 AC to 12 DC

2-Convert the AC voltage into DC:

- using Rectifier (Diode) to convert AC to DC
- using Capacitor as a filter
- voltage divider to reduce the voltage applied on MCU

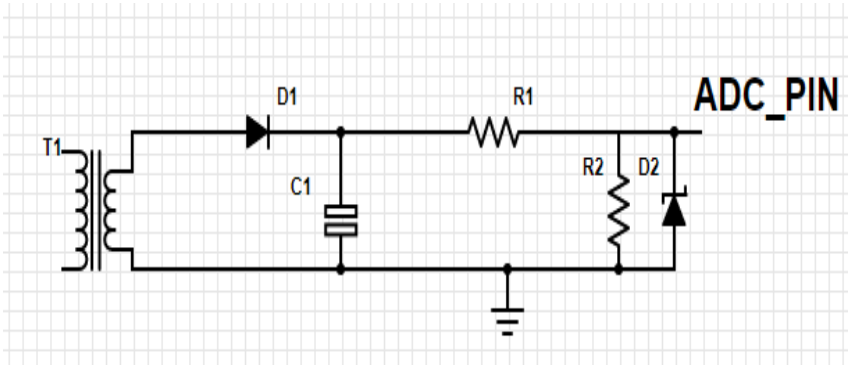


Figure 2-16 Voltameter circuit

- calculate values of resistors and capacitors

For ideal mode in atmega32, I should be 1mA

Target to get V_o = 5v (max volt to MCU)

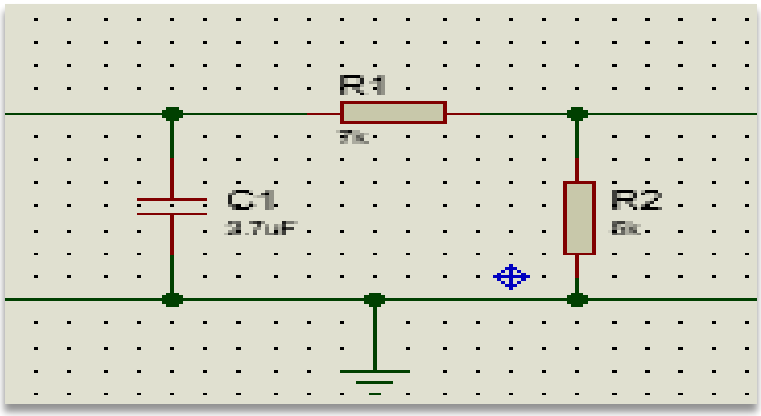


Figure 2-017 Filter + Voltage Divider Part in Voltameter Circuit

V_{in} = 12v, I_{mcu}= 1mA.

$R_2 = \frac{5}{1mA}$ = 5 KΩ

V= IR

$(R_1+R_2) = \frac{V}{I} = \frac{12}{1mA}$ = 12 KΩ

R_1= 7 KΩ

$X_c = \frac{V}{I_c}$, I_C + R_c = 0, I_C=1mA

$X_c = \frac{12}{1mA}$ =12 KΩ

$X_c = \frac{1}{W_C}$

C = $\frac{1}{2\pi*50*12000}$ = 2.653 Uf

3-Give it to the microcontroller for Calculation

- using ADC which converts the physical quantity (continuous in value) to a digital code (discrete value) to can process this quantity and control the physical world.

2.3.2: Measure Current

- using a current sensor (ACS712 Current Sensor Module) to sense current and then measure it using ADC can process this quantity and control the physical world.

2.3.3: Measure ∅

- The power factor is the "angle cosine of that lagging current "current is lagging by voltage with some angle and if take the cosine of that angle we will get the power factor: **Power factor = Cos (θ)**. So, we need to develop a method to measure the phase angle difference between voltage and current waveform.

- We here measure the Power factor using the zero-cross detection method:

  It is a method used for AC power control circuits whenever a sine wave crosses from a positive cycle to a negative cycle or a negative cycle to a positive cycle. enable us to measure the time between voltage and current.

  with the help of a simple operational amplifier as a comparator (LM358) to detect zero crossing of the sine wave by comparing high value (one)when zero-crossing occurs with low value (0).

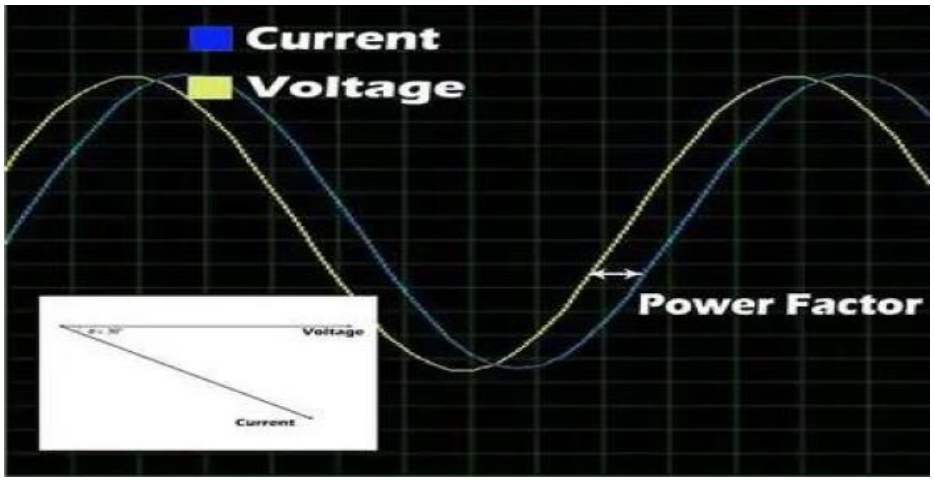


Figure 2-18 Current and Voltage Waveforms

- two zero-crossing detection circuits are used:

  1- one for the voltage waveform

  2- the other is for the current waveform.

- After detecting zero-crossing, measure the time difference between zero crossing of voltage and current waveforms using timer0 in atmaga32.

- Using interrupt pin and Timer to measure several counts:

  1- first external interrupt when detecting zero crossing of current wave and timer start count.

  2- Second external interrupt when detecting zero crossing of voltage wave and timer stop count.

3- Timer value =time difference between two waves.

4- covert it into second

t= average value of timer/1000000(timer freq)

5- Find our required angle

θ= 360 *freq(60)*t

Power factor = Cos (θ)

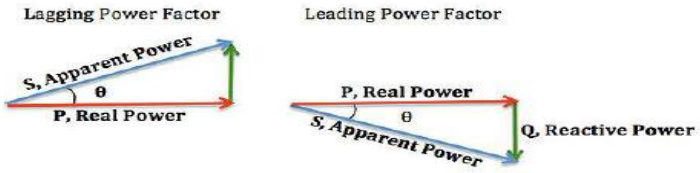


Figure 2-19 Phase Shift Between Real Power and Apparent Power

2.3.4: Calculate the energy

$$P = V \times I \times \cos \emptyset$$

a. We have calculated the Voltage, Current, and cos(φ) as well.

So, we can easily measure the power by multiplying these three parameters.

2.3.5: Calculate energy consumption

energy consumed = P*t

b. to calculate the unit consumed we must multiply the power by the time (t) seconds then we can get the unit consumed. (1unit =1 KWh)

2.4: Simulation Results

2.4.1: Voltameter Testing

- When amplitude = 311, expected: 220

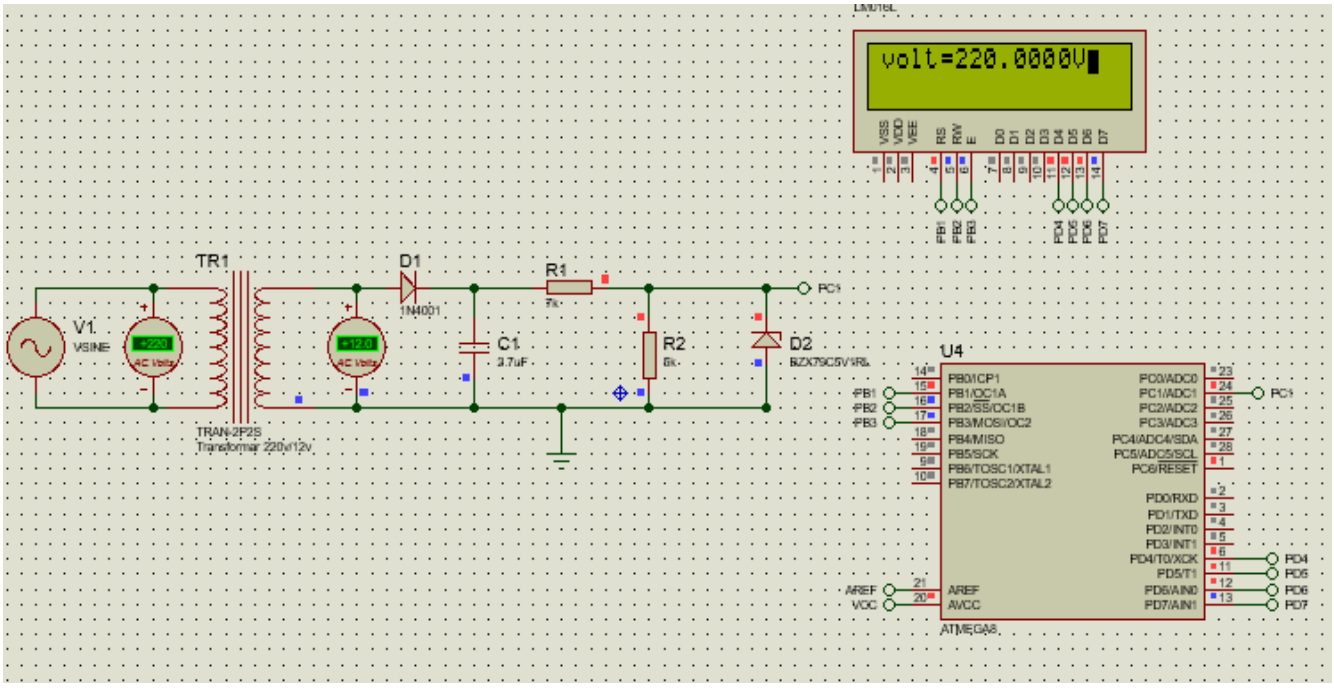


Figure 2-20 Voltameter Testing When Amplitude= 311

- When amplitude = 200, expected: 142

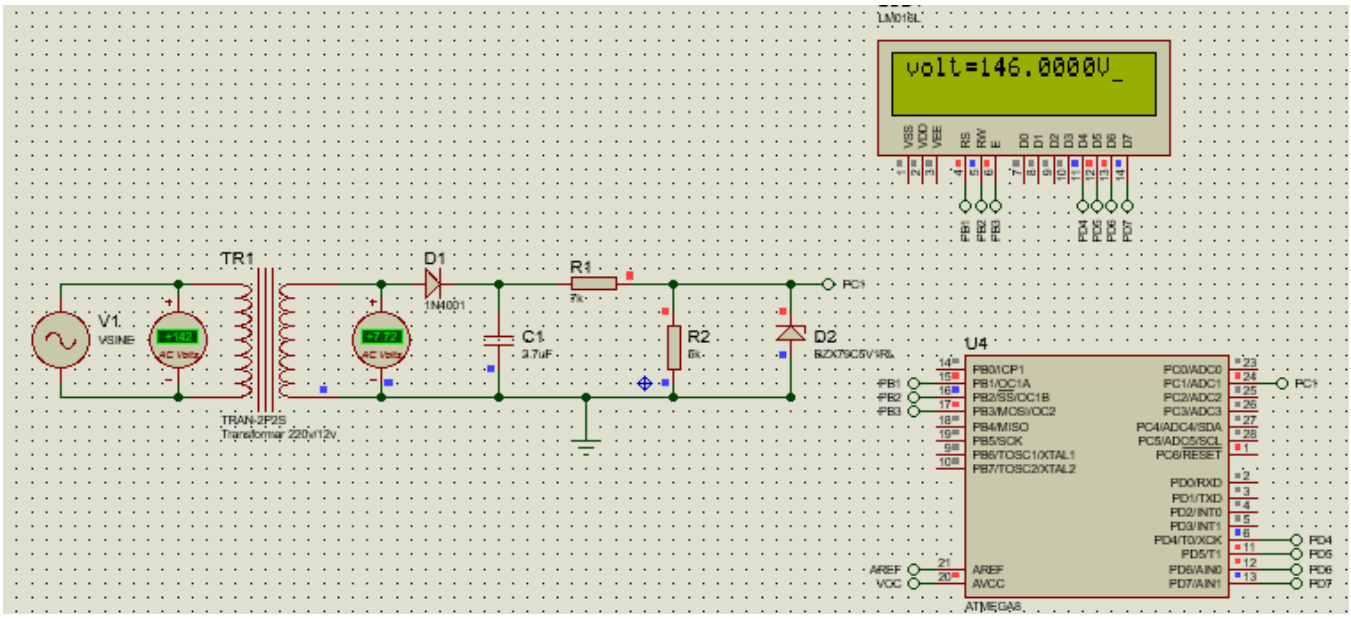


Figure 2-21 When Amplitude= 200

2.4.2: Ammeter Testing

- When amplitude = 311

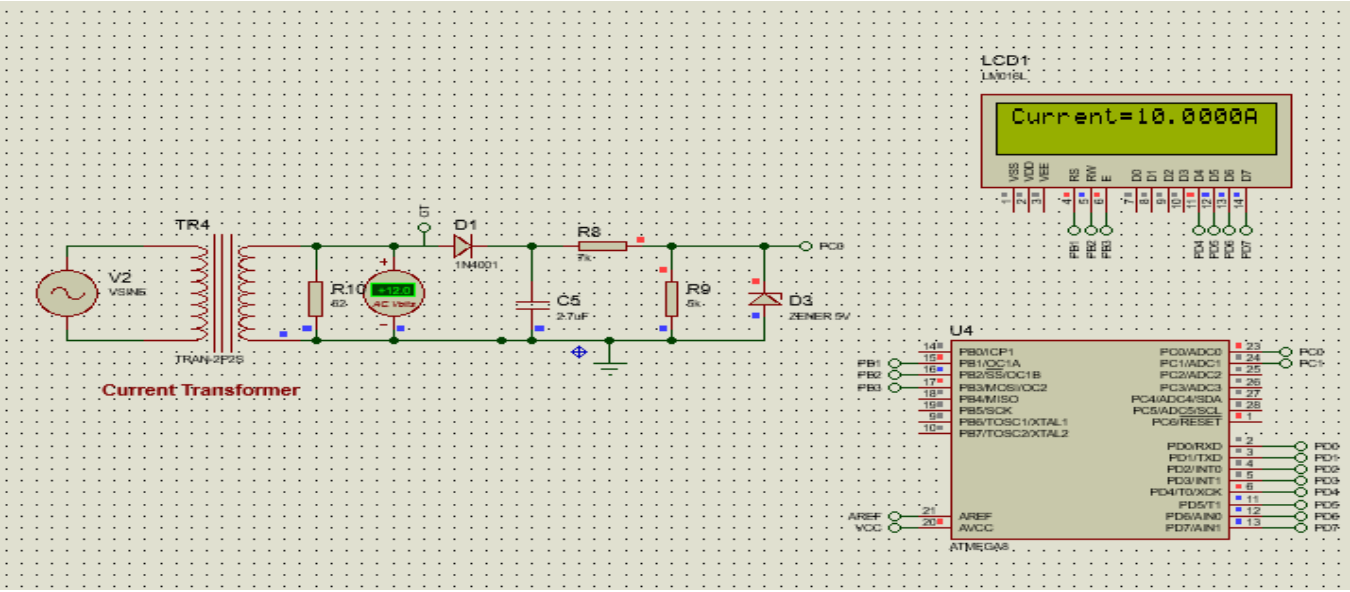


Figure 2-22 Ammeter Testing When Amplitude= 311

- When amplitude= 200

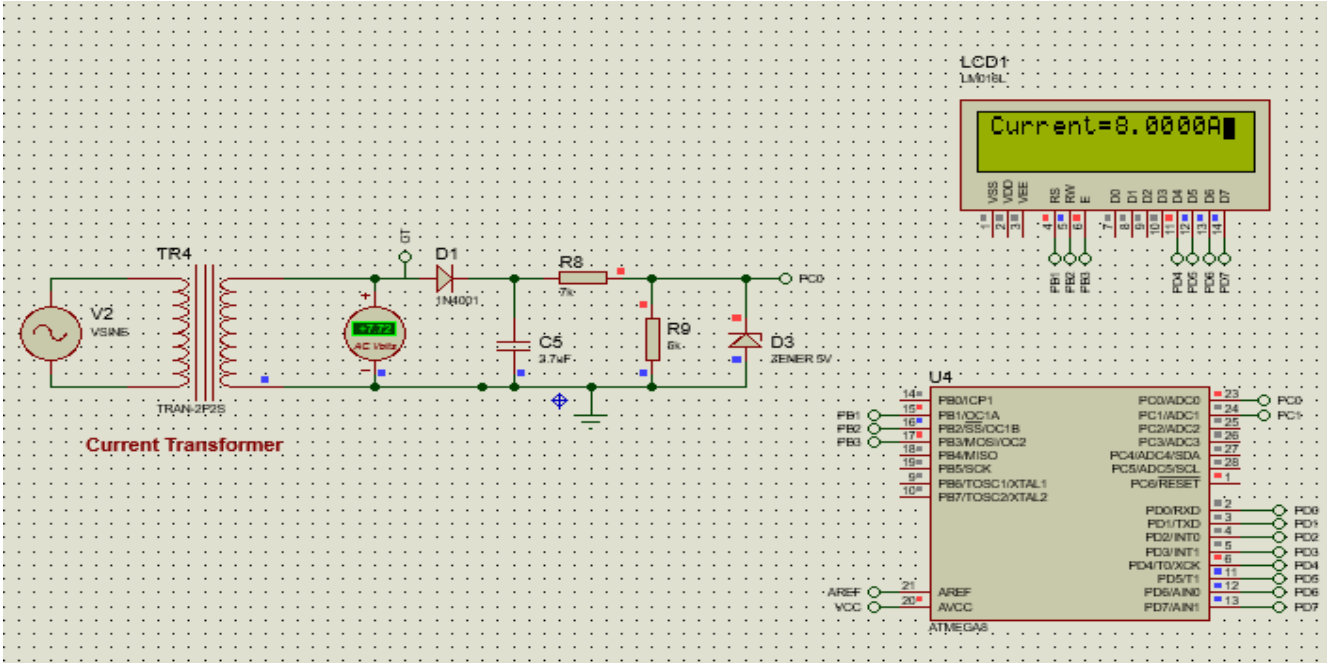


Figure 2-23 Ammeter Testing When Amplitude= 200

2.4.3: Power Factor Testing

- When phase between voltage and current = 0
- expected value: 1
- The power factor is 0.9973 which is very good. it means that our system's efficiency is 99.73% and the energy dissipation is also 99.73% so our system's efficiency is as well 99.73%.

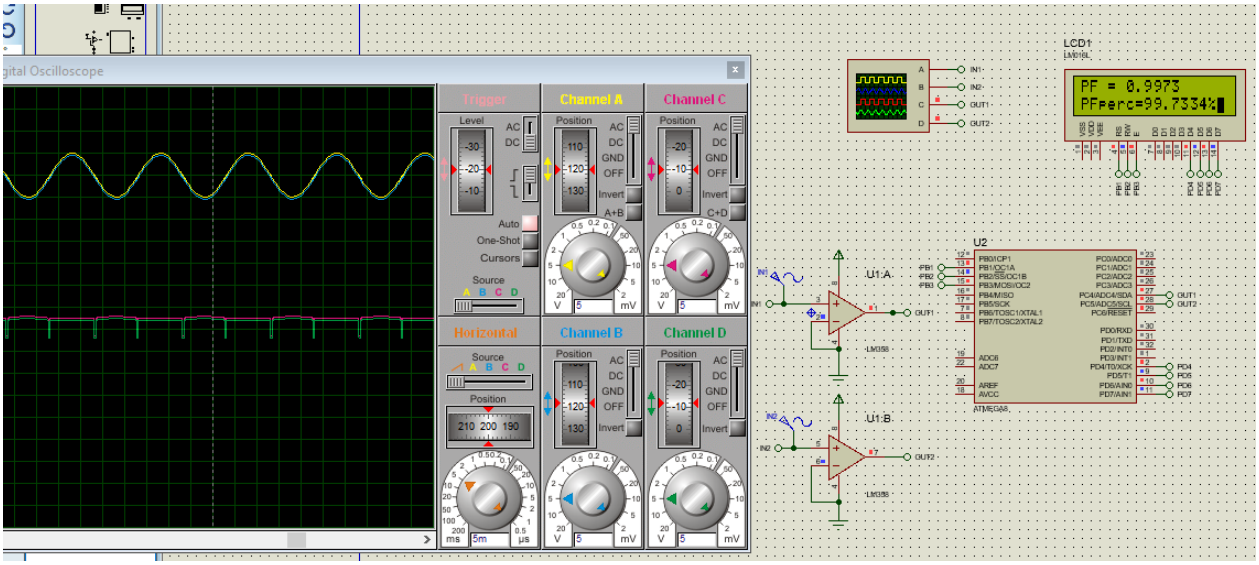


Figure 2-24 Power Factor Testing When phase =0

- When phase between voltage and current = 30
- expected value: 0.866
- power factor is 0.8659. it means that our system's efficiency is 86.59%

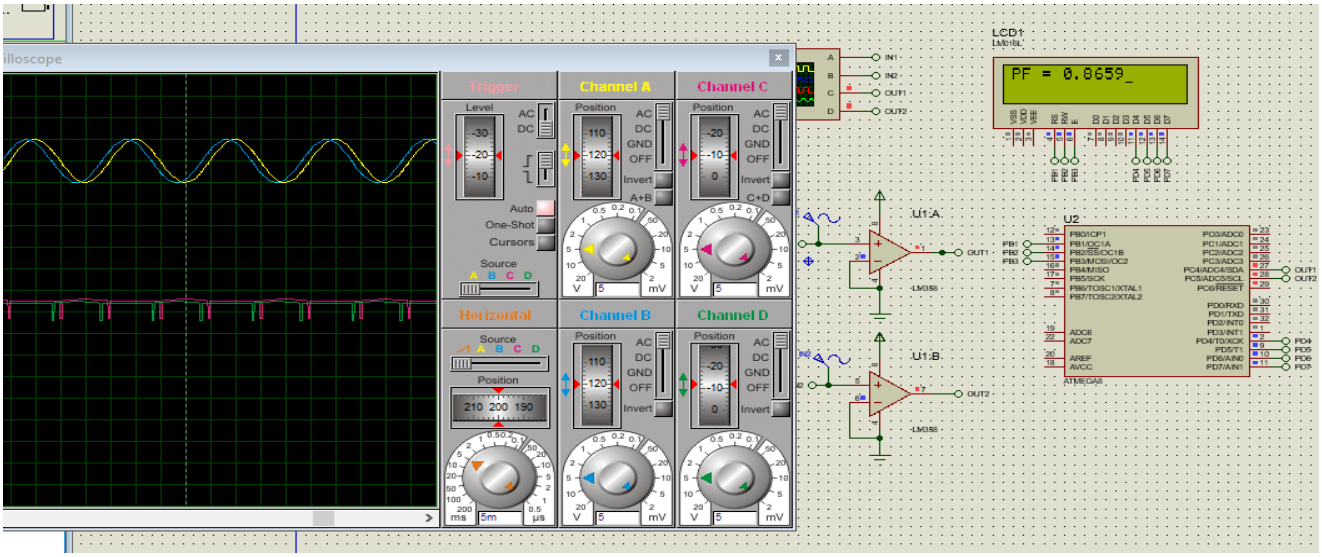


Figure 2-25 Power Factor Testing When phase =30

- When the phase between voltage and current = 90
- expected value: 0
- power factor is 0.0047. it means that our system's efficiency is 0.47%. of all power losses.

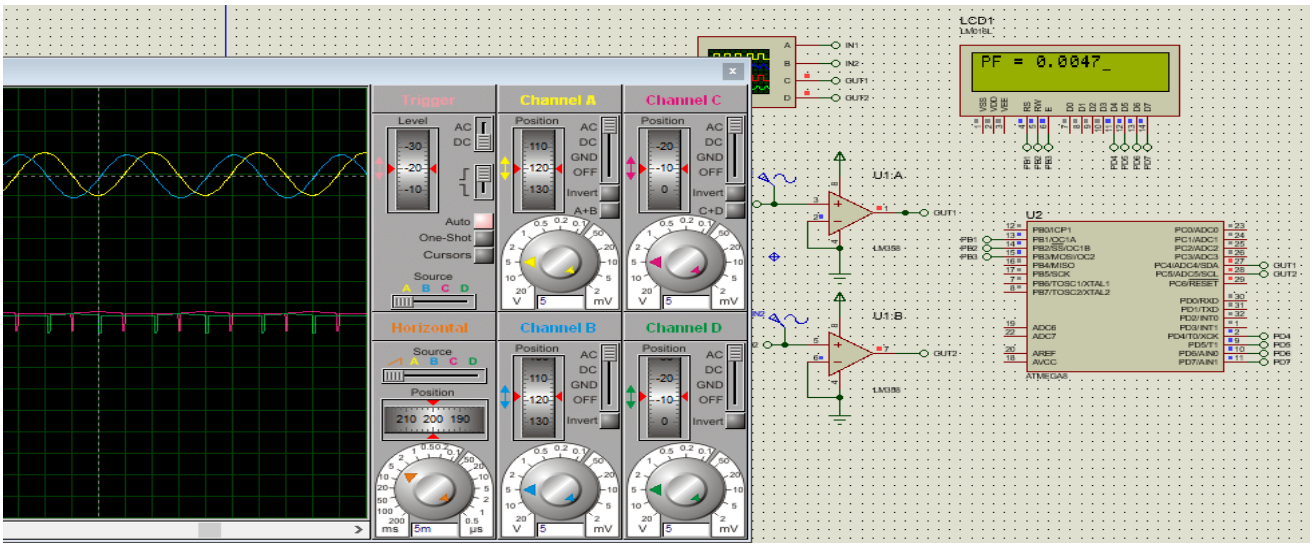


Figure 2-26 Power Factor Testing When phase =90

2.5: Smart Meter Circuit

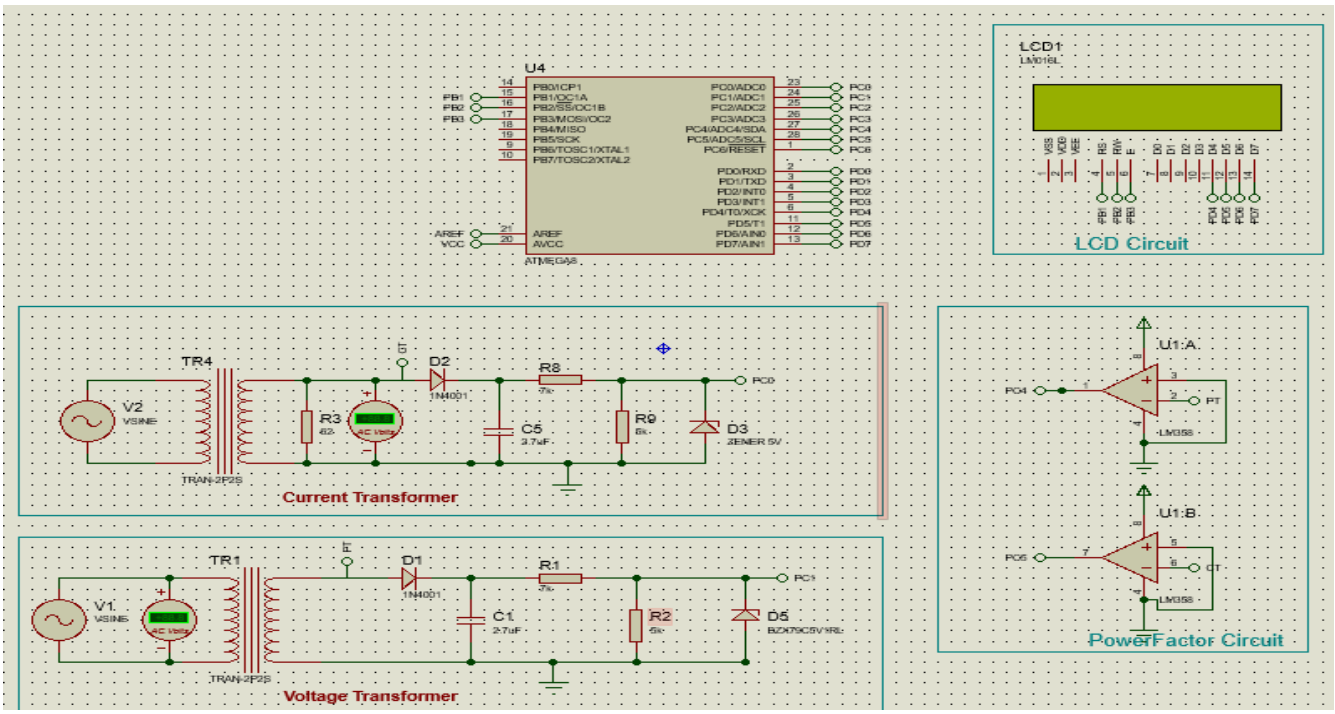


Figure 2-27 Smart Meter Circuit

3. Chapter 3: Smart Lighting System

3. 1 Overview

In this chapter, we will solve the problem of accidents occurring in the nighttime is either caused by driver errors or malfunctioning streetlights The malfunctions can be caused by fused bulbs or damaged wiring. This project reduces this error hence reducing accidents significantly.

Also, Manual control is prone to errors and leads to energy wastage, and manually controlling during midnight is impracticable. Where the Smart Street Lighting System is an advanced lighting system that is designed to increase the accuracy and efficiency of the streetlight by timed controlled Switching.

3.2. System Analysis & Design

3.2.1. Block diagram

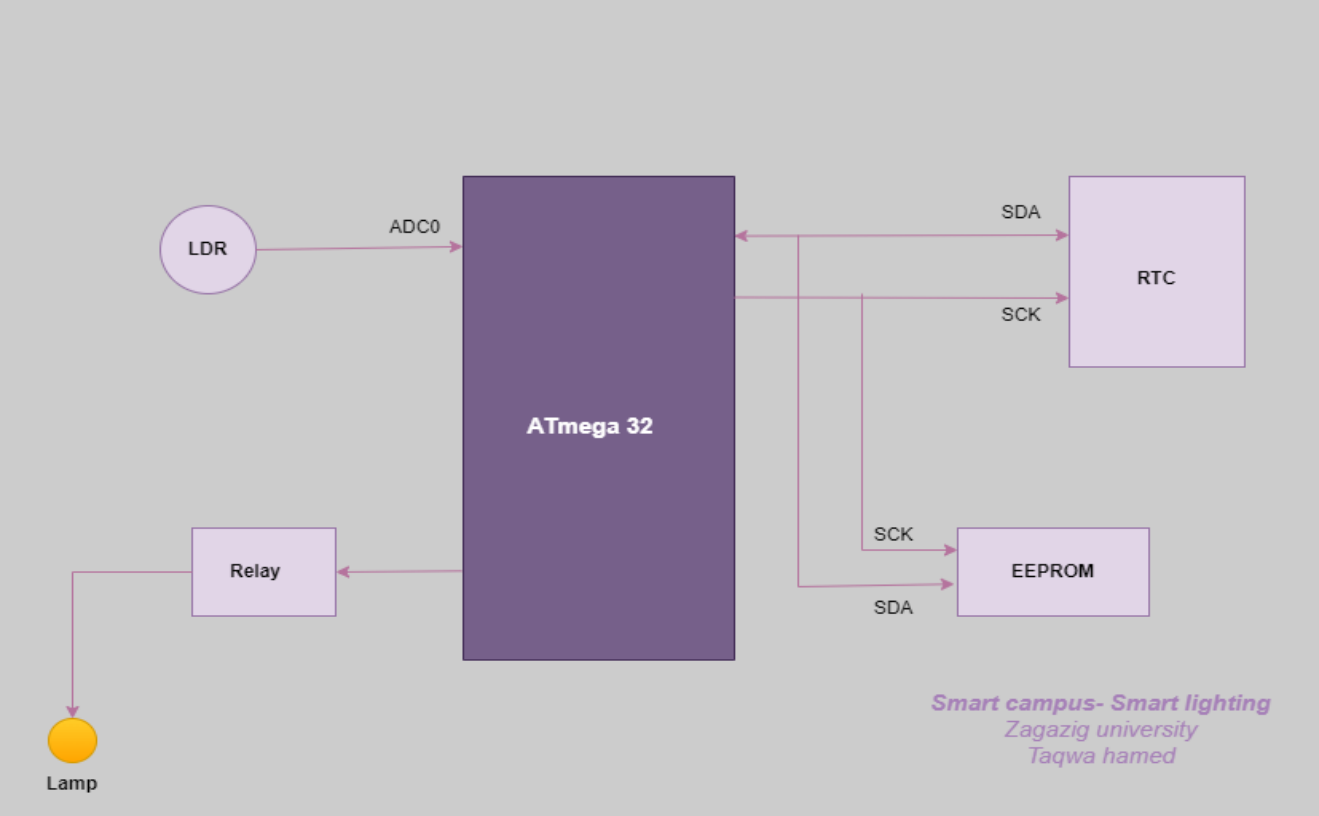


Figure 3-1 The System Architecture

3.2.2. **Flow chart**

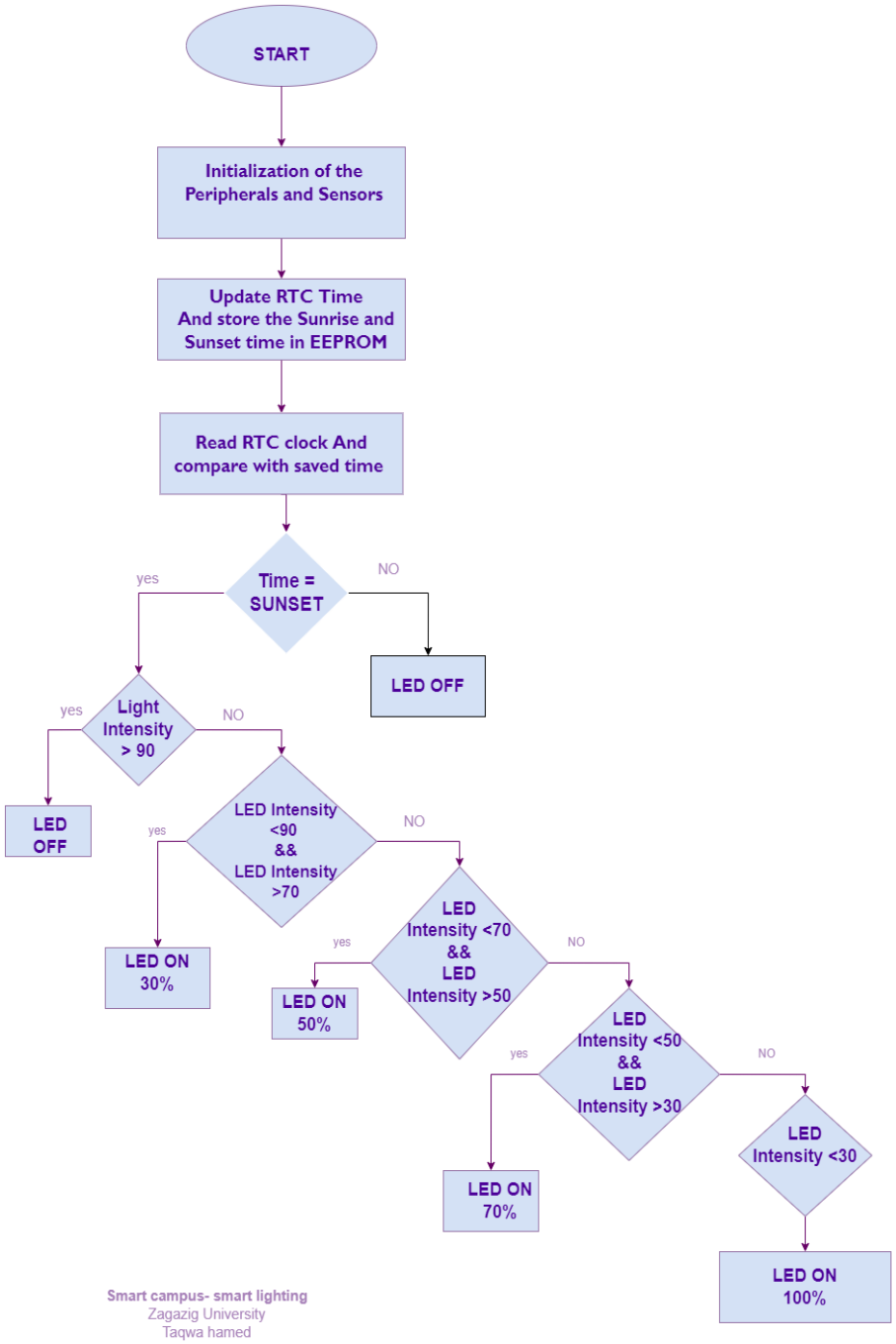


Figure 3-2 System Flow Chart

3.2.3. **Pseudocode**

- First, initialize peripherals & sensors.
- Second update RTC time.
- Third Store sunrise & sunset time in EEPROM.

❖ **Loop** Forever

- **IF** Time = sunset

First check **IF** light intensity > 90

LED OFF

Else

IF light intensity < 90 && light intensity > 70
LED ON 30 %
Else
check **IF** light intensity < 70 && light intensity > 50
LED ON 50 %
Else
check **IF** light intensity < 50 && light intensity > 30
LED ON 70 %
Else
check **IF** light intensity < 30
LED ON 100 %

Else

LED OFF

3.1.1. Sequence Diagram

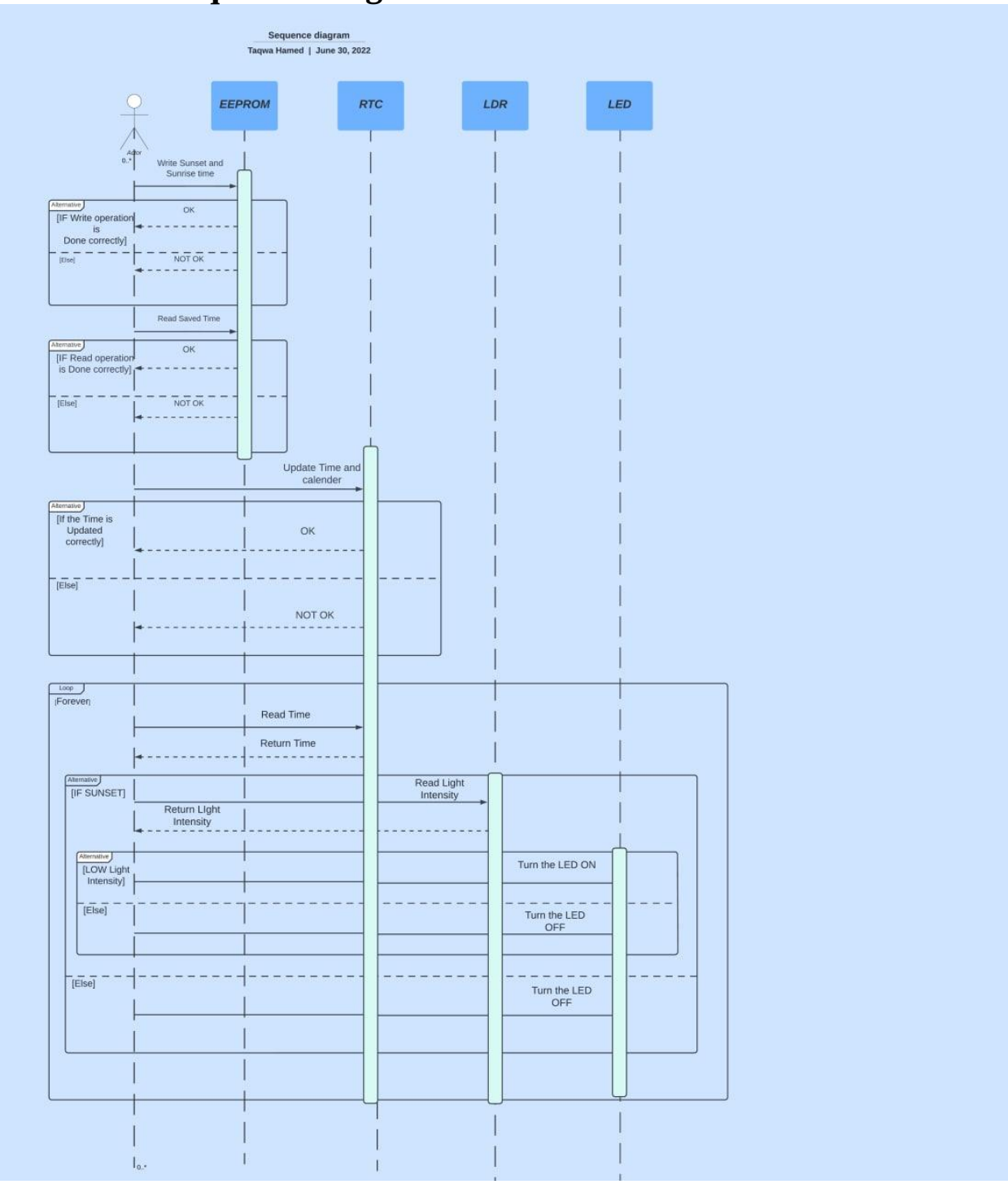


Figure 3-3 Sequence Diagram

3.2.5. **Hardware tools**

3.2.5.1. **ATMEGA 32**

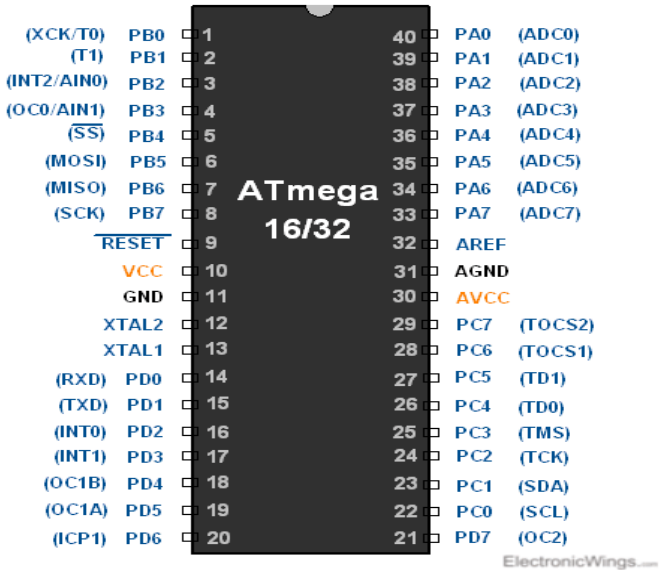


Figure 3-4 Microcontroller Pinout

The Atmel®AVR®ATmega32 is a low-power CMOS 8-bit microcontroller based on the AVR-enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed**.**

3.2.5.2. **LDR SENSOR**

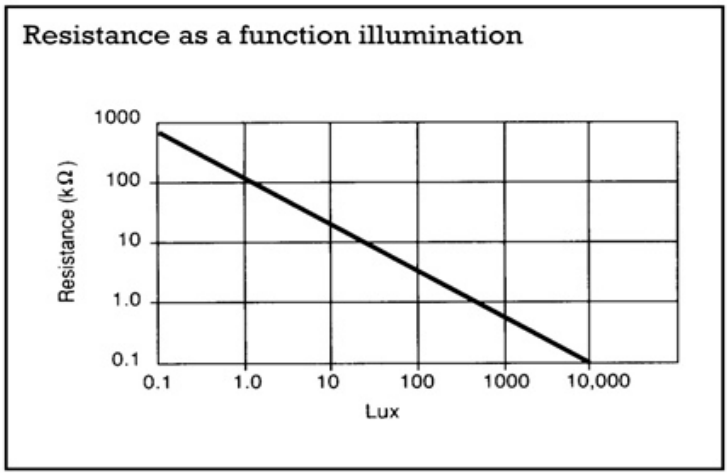


Figure 3-5 LDR Characteristics

- Measures the intensity of the external light and when the degree of darkness is high, the LEDs go to an ON state at night. (When the light intensity is increased the resistivity is increased hence the voltage drops).
- This system works by sensing the intensity of light in its environment. The sensor that can be used to detect light is an LDR.
- The LDR gives out an analog voltage when connected to VCC (5V), which varies in magnitude in direct proportion to the input light intensity on it. That is, the greater the intensity of light, the greater the corresponding voltage from the LDR will be.

3.2.5.3. **ADC**

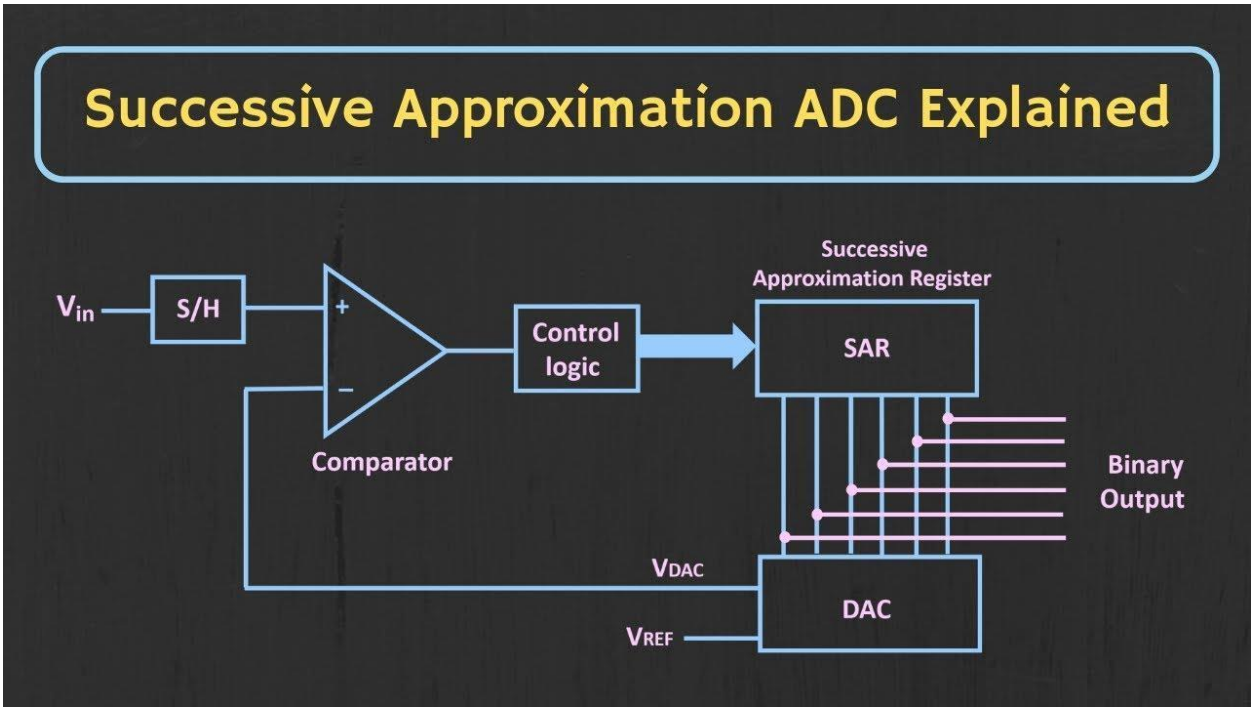


Figure 3-6 Analog-To-Digital Converter Hardware Circuit

Successive Approximation ADC (SAR ADC)

Its basic idea depends on a counter that counts on a clock signal called ADC Clock and with every count it increases a compare analog signal voltage value that is compared to the input signal, the counter stops counting when the compared signal is approximately equal to the input signal. Then the counter value represents the digital value of the input signal.

This graph shows the mapping between the analog values and digital values for ***3 bit ADC*** with reference voltage = 5V.

The 3 bit ADC has 8 possible digital values. So, simply the step = 8 / 5 = **0.625v**.

Important Note:
The ADC saturates at voltage =
Reference Voltage – 1 step.

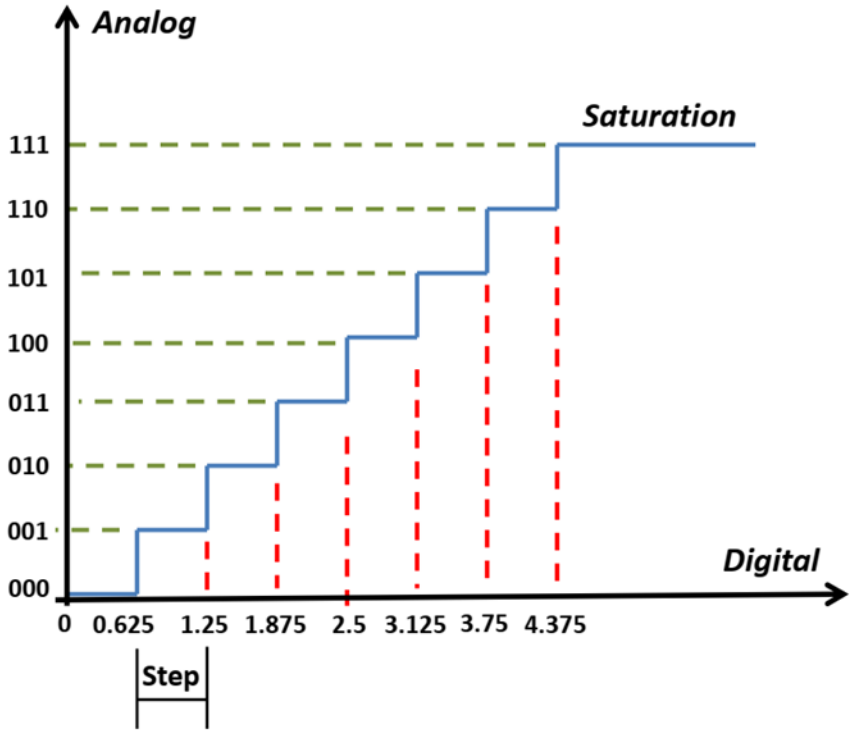


Figure 3-7 ADC Quantization levels

3.2.5.4. **PWM**

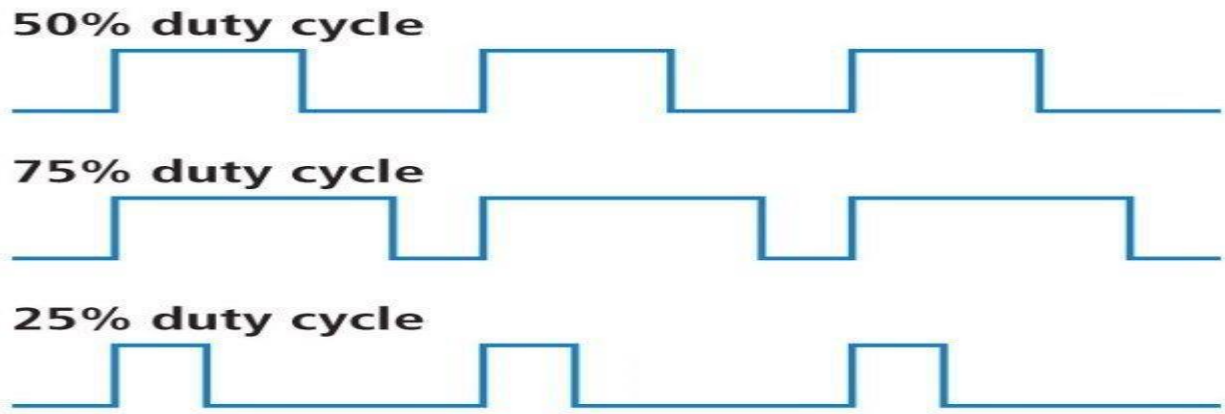
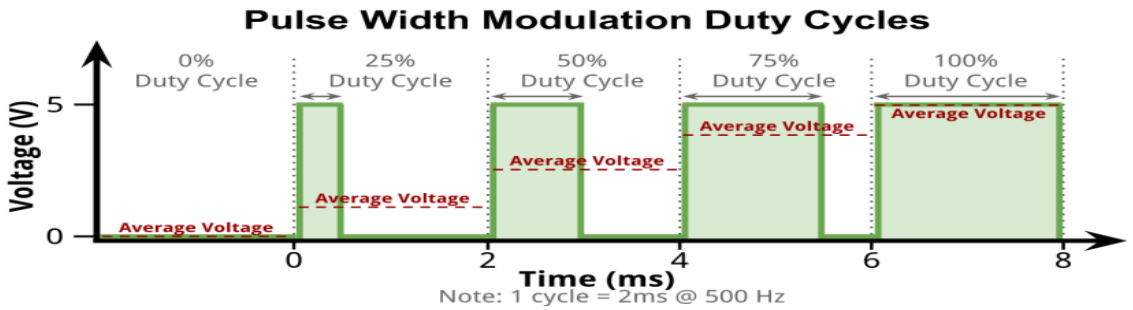


Figure 3-8 PWM With Different Duty cycles

- Pulse Width Modulation (PWM) is a method for changing how long a square wave stays "on".
- The on-off behavior changes the average power of the signal.
- If the signal toggles between on and off quicker than the load, then the load is not affected by the toggling.

- The duty cycle is a percentage measurement of how long the signal stays on.
- *The effective voltage* of the PWM signal is called Root Mean Square (RMS) which equals to:

RMS = Amplitude $\sqrt{\text{Duty Cycle}}$



The AVR already has a PWM modes

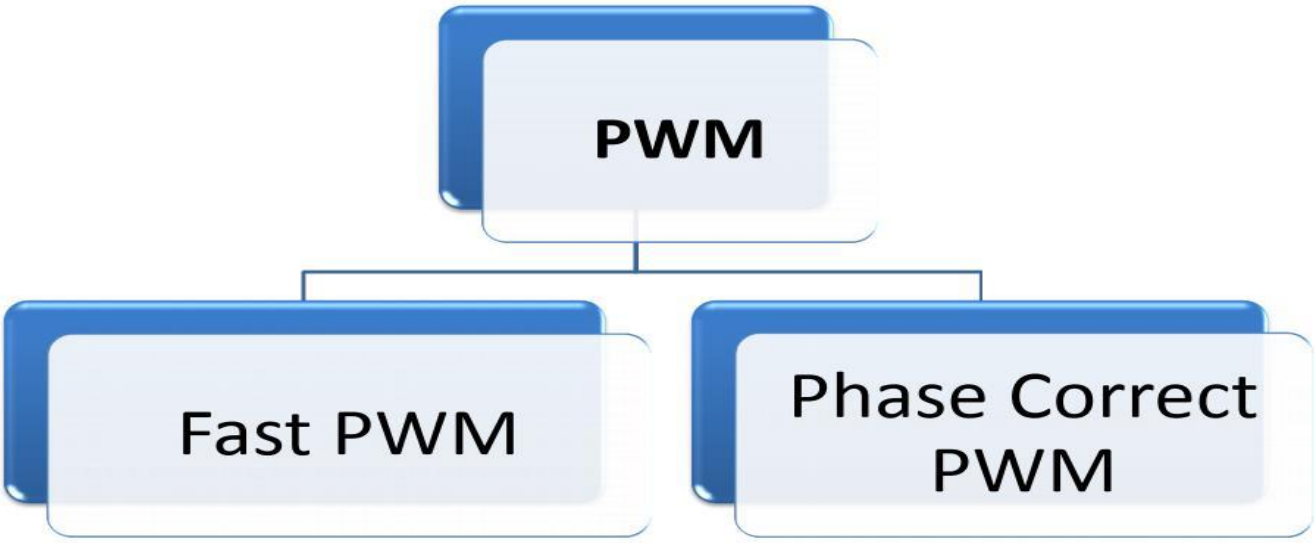


Figure 3-9 Types of PWM in Microcontroller

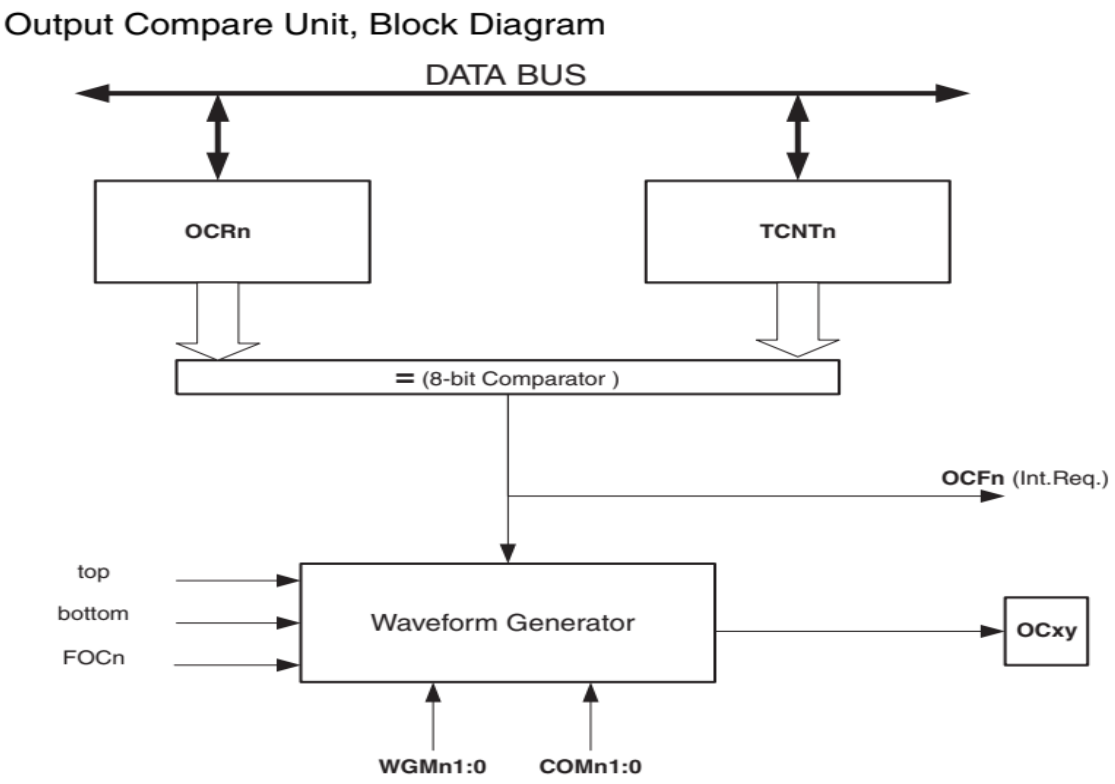


Figure 3-10 PWM Circuit

❖ Fast PWM:

Figure 58. Fast PWM Mode, Timing Diagram

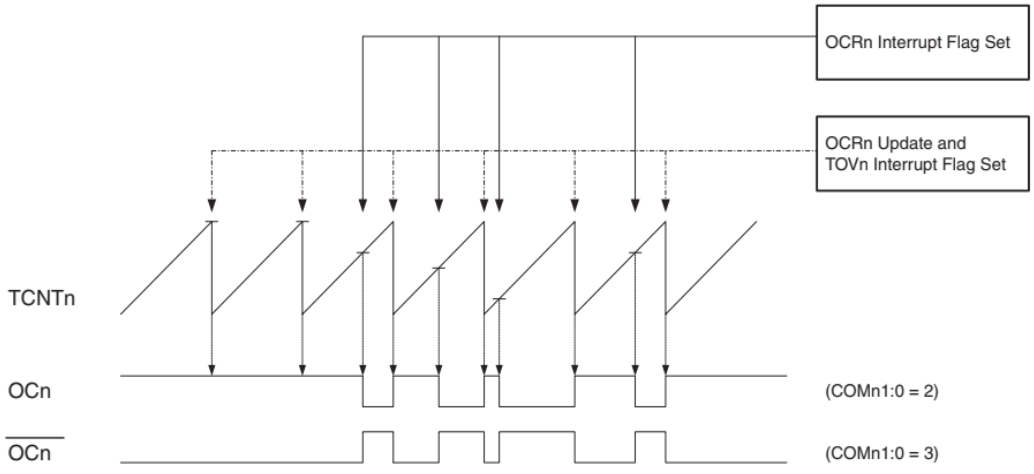


Figure 3-11 Fast PWM

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

❖ Phase Correct PWM:

PWM frequency for the output when using phase correct PWM can be calculated by the following equation:

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

The N variable represents the prescale factor (1, 8, 32, 64, 128, 256, or 1024).

Figure 47. Phase Correct PWM Mode, Timing Diagram

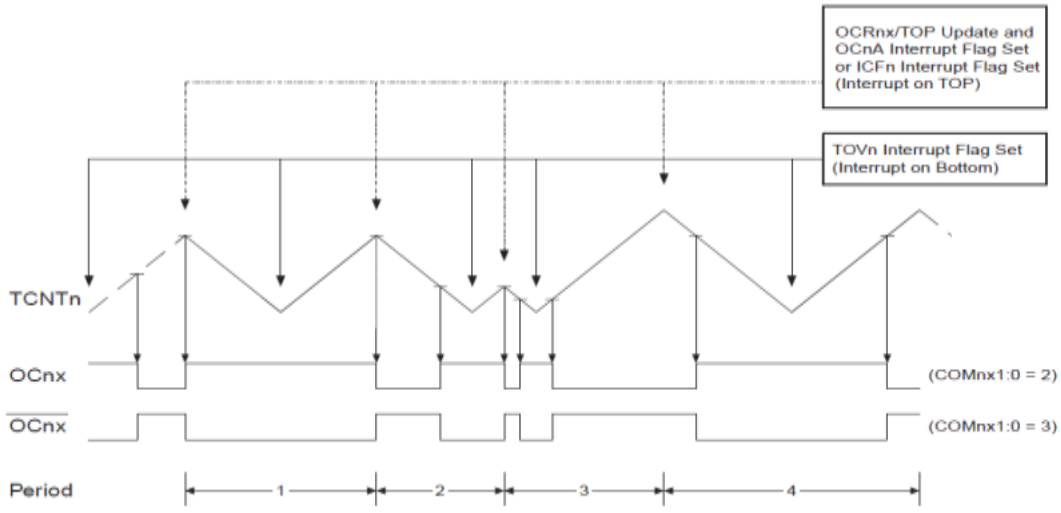


Figure 3-12 Phase Correct PWM

3.2.5.5. I2C

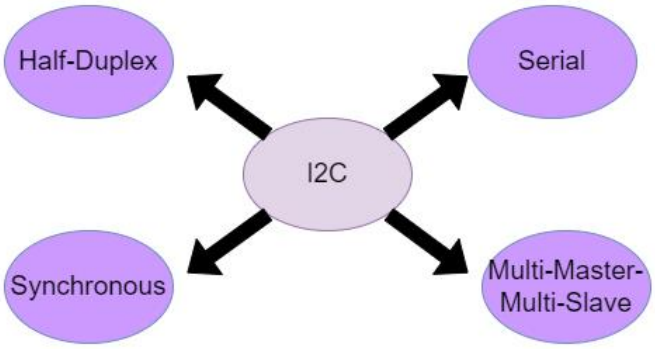


Figure 3-13 I2C Features

- Inter-Integrated Circuit I²C (Pronounced I Two C or I Squared C), is a serial communication protocol in which the devices are hooked up to the I2C bus with just two wires.
- It is sometimes referred to as a Two-Wire Interface or the TWI.
- Devices could be the CPU, IO Peripherals like ADC, or any other devices which support the I2C protocol.
- All the devices connected to the bus are classified as either being Master or slaves.

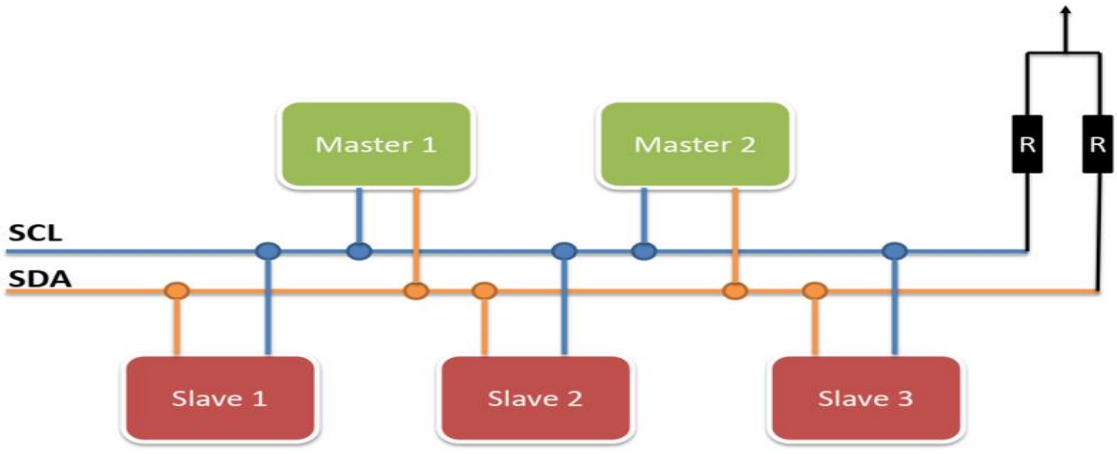


Figure 3-14 I2C Bus Structure

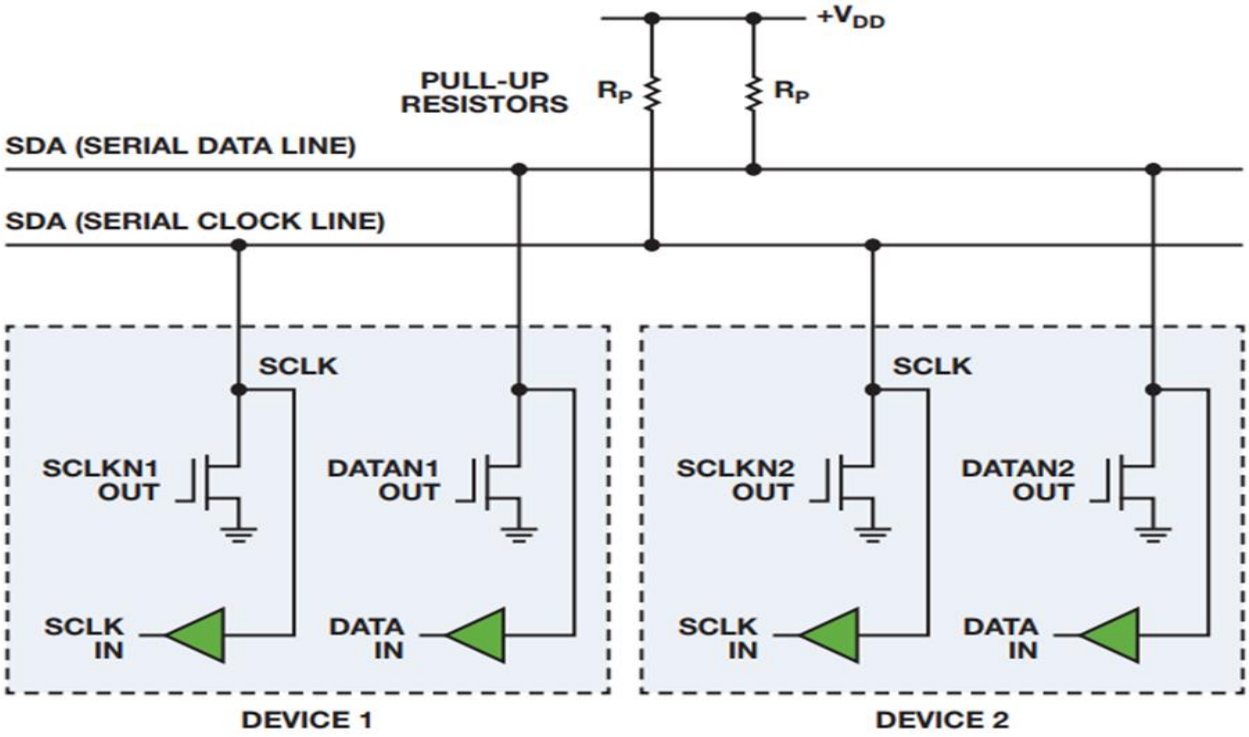


Figure 3-15 I2C Open-Drain Bus

If the bus is an open-drain line, then any node that writes 0 the bus would be derived to GND, if all nodes write 1 then the bus would be floating, that's why we use a pull-up resistor to connect the line to VCC which will be effective only if all nodes write 1.

It looks like AND gate, if any nodes write 0, the bus is derived to GND, if all nodes write 1, the bus is derived to VCC by the pull-up resistor.

- **BUS Arbitration**

Node 1	Node 2	Result on Line
0	0	GND
1	1	VCC
0	1	GND
1	0	

Figure 3-16 I2C Bus Arbitration

- **I2C Frame**

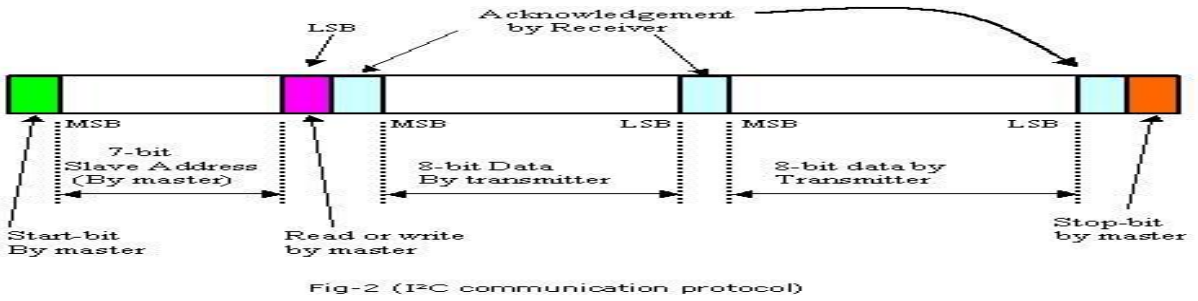


Fig-2 (I²C communication protocol)

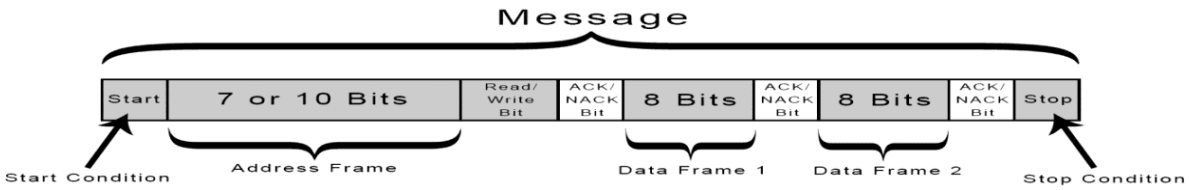


Figure 3-17 I2C Frame

3.1.2.6. **RTC Module DS1307**

64 x 8, Serial, I2C Real-Time Clock

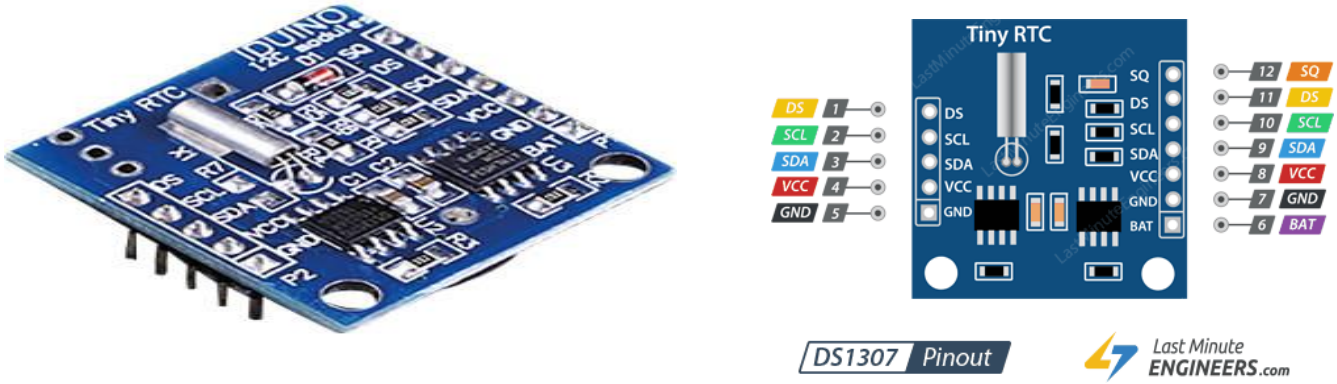


Figure 3-18 RTC Module

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, days, dates, months, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects

power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply

PIN DESCRIPTION

PIN	NAME	FUNCTION
1	X1	Connections for Standard 32.768kHz Quartz Crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (C_L) of 12.5pF. X1 is the input to the oscillator and can optionally be connected to an external 32.768kHz oscillator. The output of the internal oscillator, X2, is floated if an external oscillator is connected to X1. **Note:** For more information on crystal selection and crystal layout considerations, refer to *Application Note 58: Crystal Considerations with Dallas Real-Time Clocks*.
2	X2	
3	V_{BAT}	Backup Supply Input for Any Standard 3V Lithium Cell or Other Energy Source. Battery voltage must be held between the minimum and maximum limits for proper operation. Diodes in series between the battery and the V_{BAT} pin may prevent proper operation. If a backup supply is not required, V_{BAT} must be grounded. The nominal power-fail trip point (V_{PF}) voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAh or greater will back up the DS1307 for more than 10 years in the absence of power at +25°C. UL recognized to ensure against reverse charging current when used with a lithium battery. Go to: www.maxim-ic.com/qa/info/ul/.
4	GND	Ground
5	SDA	Serial Data Input/Output. SDA is the data input/output for the I^2C serial interface. The SDA pin is open drain and requires an external pullup resistor. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
6	SCL	Serial Clock Input. SCL is the clock input for the I^2C interface and is used to synchronize data movement on the serial interface. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}.
7	SQW/OUT	Square Wave/Output Driver. When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square-wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pullup resistor. SQW/OUT operates with either V_{CC} or V_{BAT} applied. The pullup voltage can be up to 5.5V regardless of the voltage on V_{CC}. If not used, this pin can be left floating.
8	V_{CC}	Primary Power Supply. When voltage is applied within normal limits, the device is fully accessible and data can be written and read. When a backup supply is connected to the device and V_{CC} is below V_{TP}, read and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage.

Figure 3-19 RTC Pin Description

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

Figure 3-20 RTC Register Description

1. Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave's address. Next follows several data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.
2. Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting several data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

The master device generates all the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

- ✓ The DS1307 can operate in the following two modes:
  1. Slave Receiver Mode (Write Mode): Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. The hardware performs address recognition after reception of the slave address and direction bit (see Figure 3-21). The slave address byte is the first byte received after the master generates the START condition. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/W), which for a write is 0. After receiving and decoding the slave address byte, the DS1307 outputs an acknowledge on SDA. After the DS1307 acknowledges the slave address + writes a bit, the master transmits a word address to the DS1307. This sets the register pointer on the DS1307, with the DS1307 acknowledging the transfer. The master can then transmit zero or more bytes of data with the DS1307 acknowledging each byte received. The register pointer automatically increments after each data byte is written. The master will generate a STOP condition to terminate the data write.

2. Slave Transmitter Mode (Read Mode): The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. The DS1307 transmits serial data on SDA while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (see Figure 5). The slave address byte is the first byte received after the START condition is generated by the master. The slave address byte contains the 7-bit DS1307 address, which is 1101000, followed by the direction bit (R/W), which is 1 for a read. After receiving and decoding the slave address the DS1307 outputs an acknowledge on SDA. The DS1307 then begins to transmit data starting with the registered address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The register pointer automatically increments after each byte is read. The DS1307 must receive a Not Acknowledge to end a read.

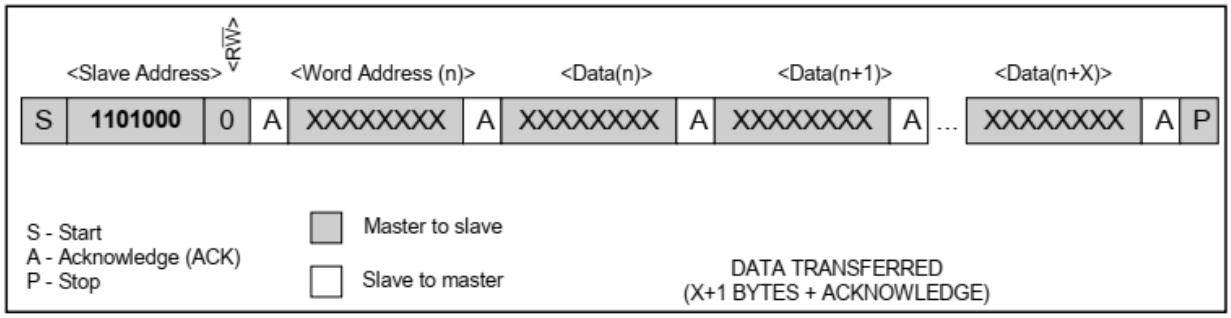


Figure 3-21 Data Write Slave Receive mode

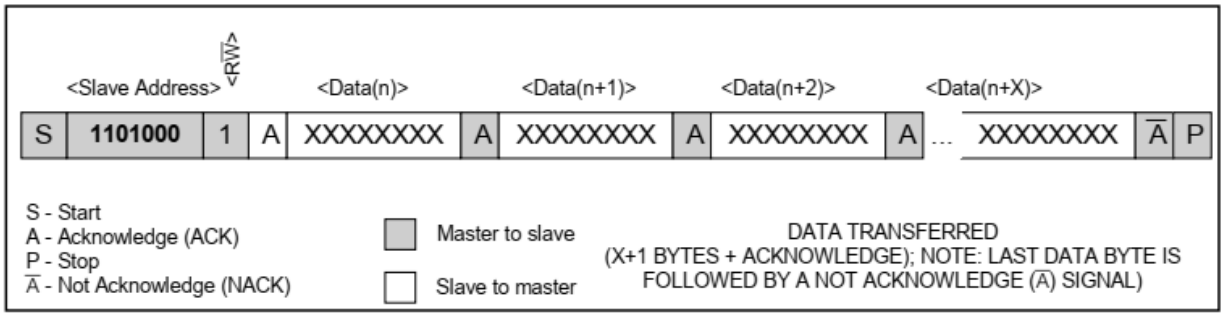


Figure 3-23 Data Read Slave Transmitter mode

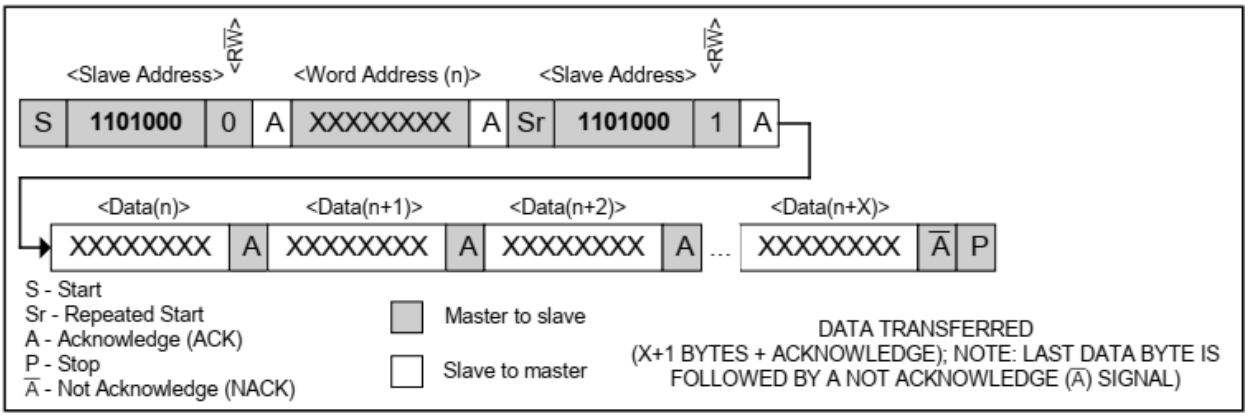


Figure 3-22 Data Read (Write Pointer, Then Read)—Slave Receive and Transmit

3.3. Simulation Results

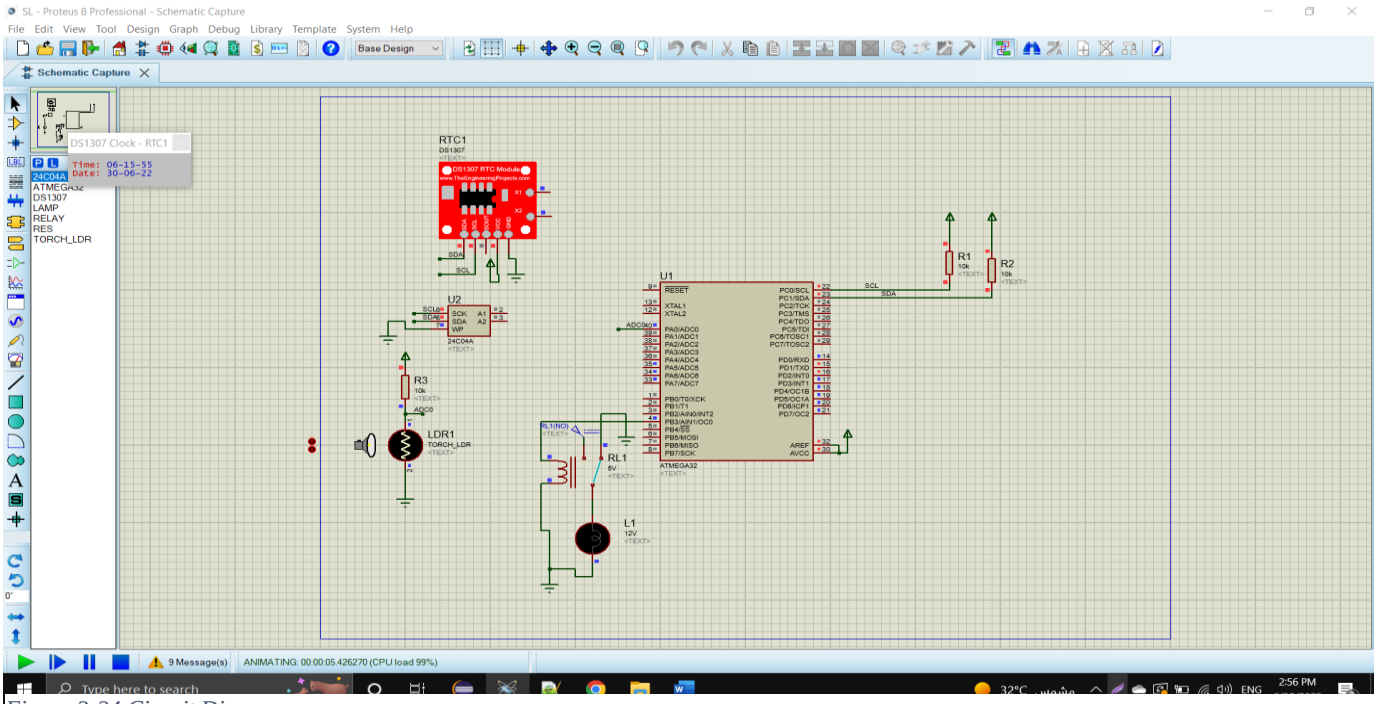


Figure 3-24 Circuit Diagram

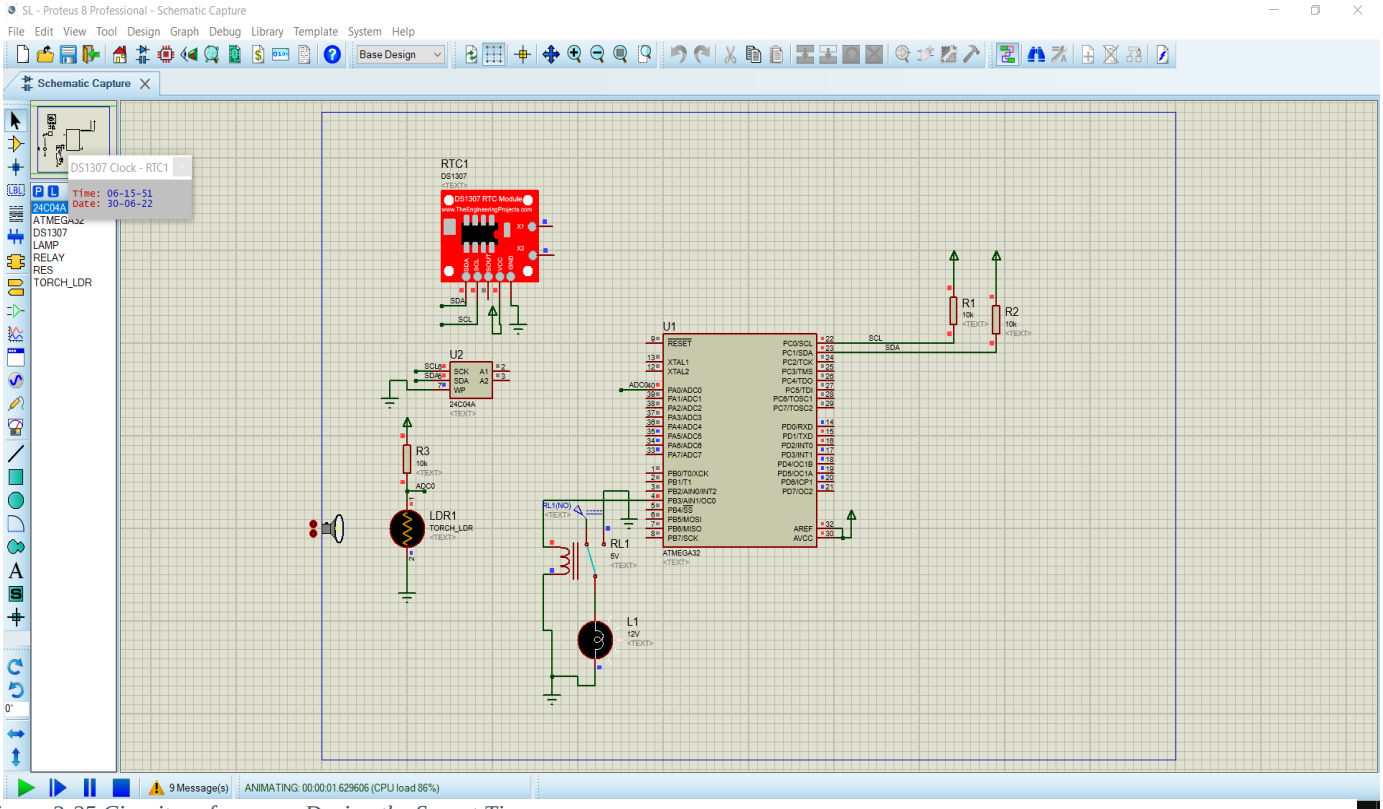


Figure 3-25 Circuit performance During the Sunset Time

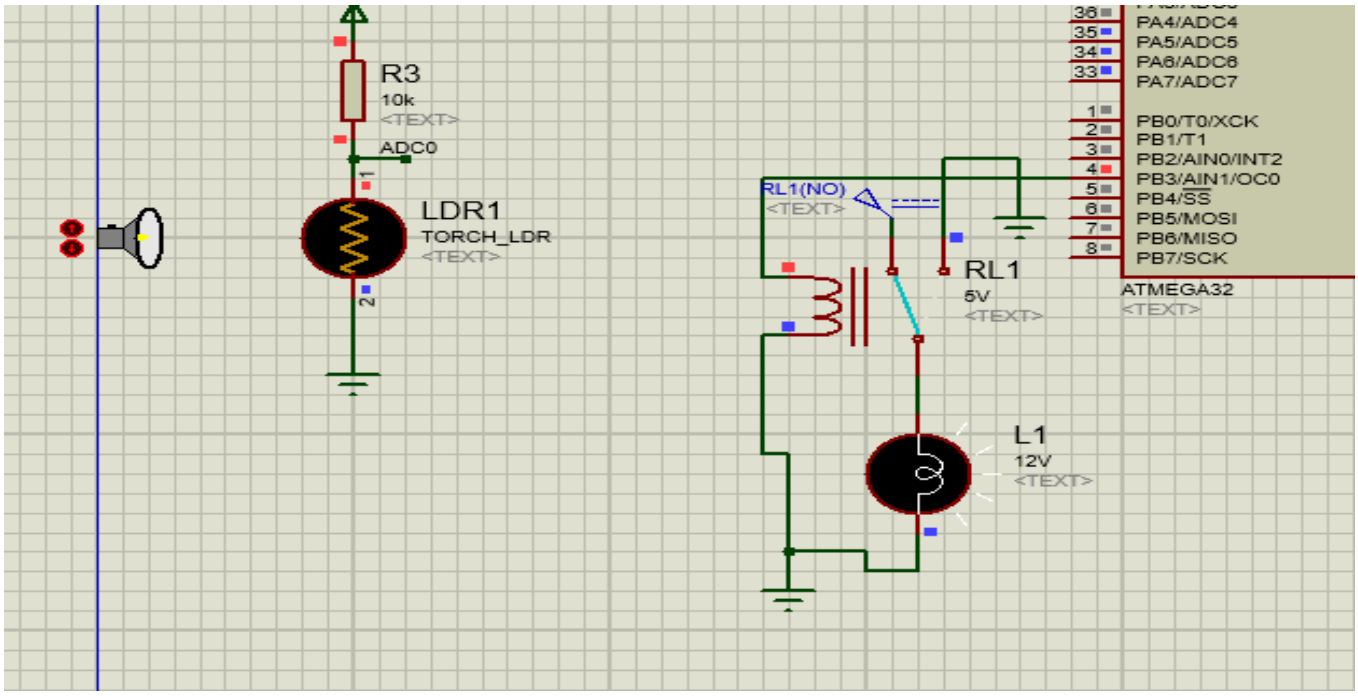


Figure 3-27 Circuit Performance During the Sunset and Low light intensity detected by LDR

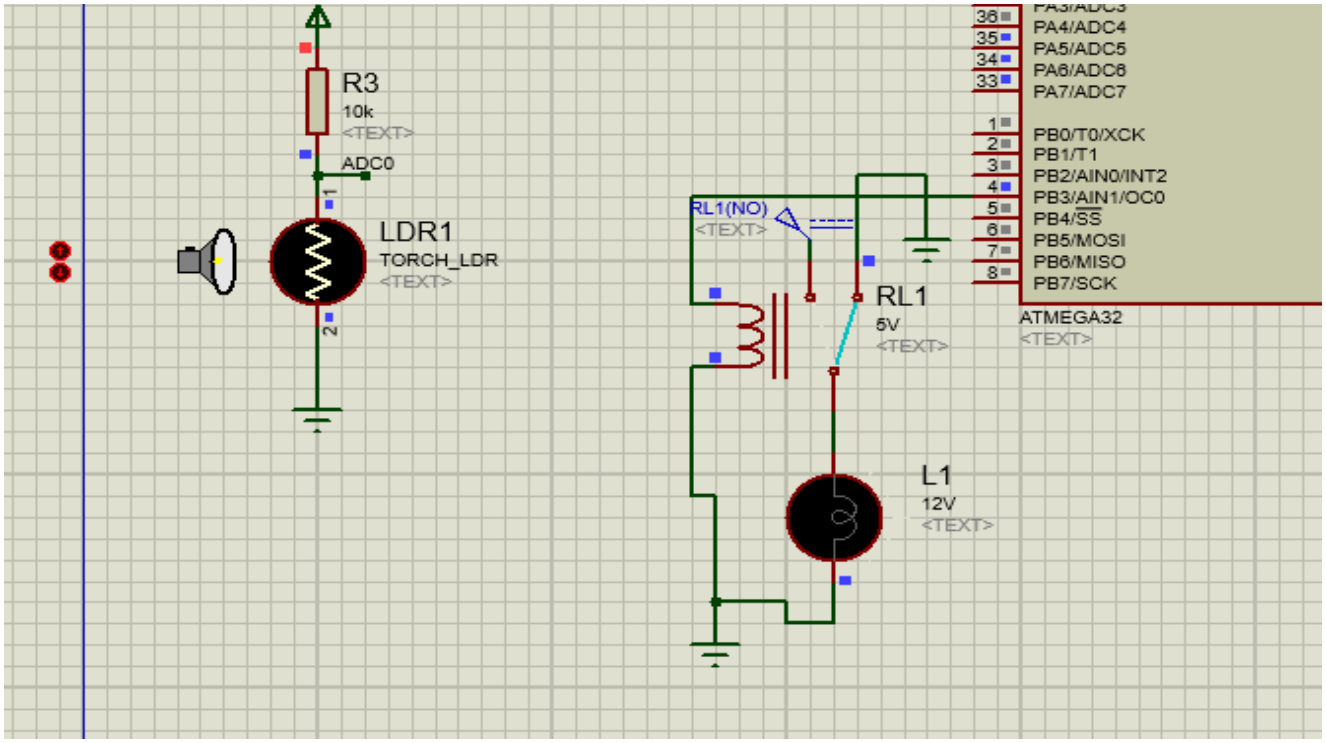


Figure 3-26 Circuit Performance During the Sunset and high light intensity detected by LDR

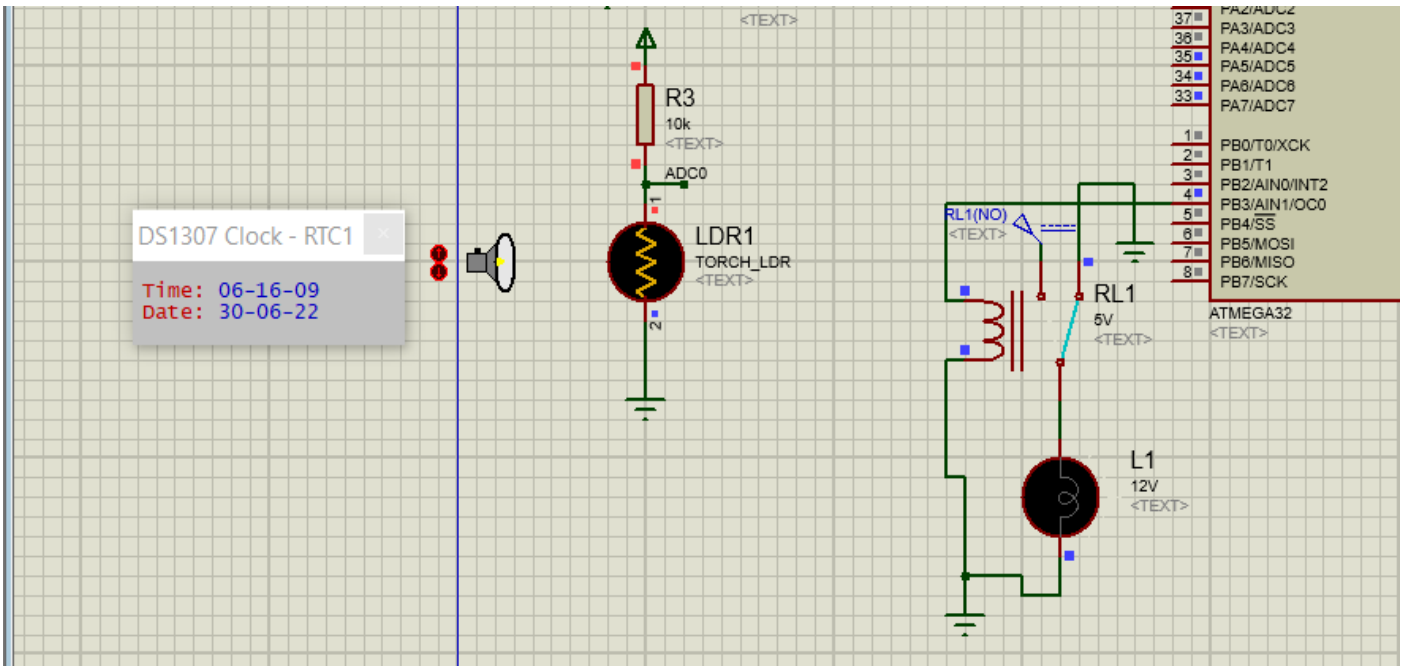


Figure 3-28 Circuit Performance during the sunrise time

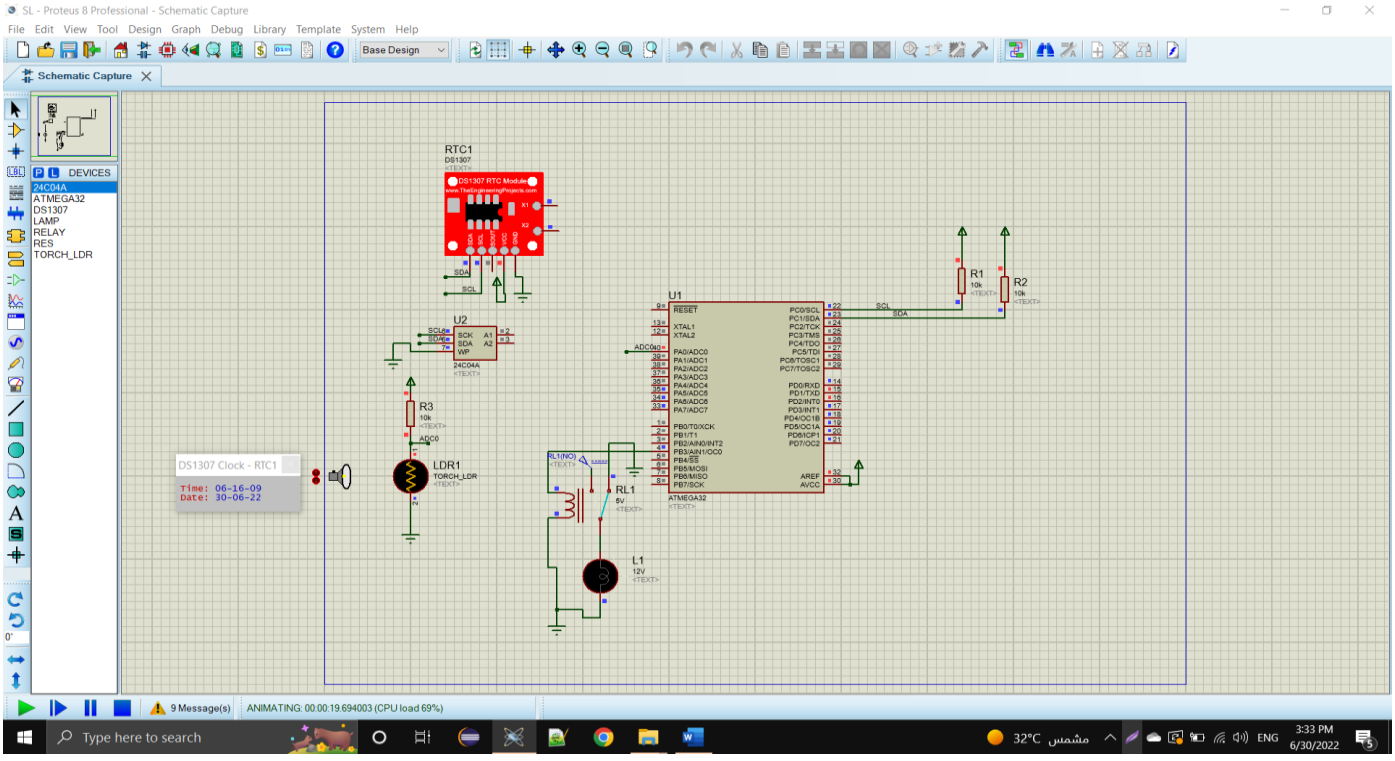


Figure 3-29 Circuit Performance during the sunrise time

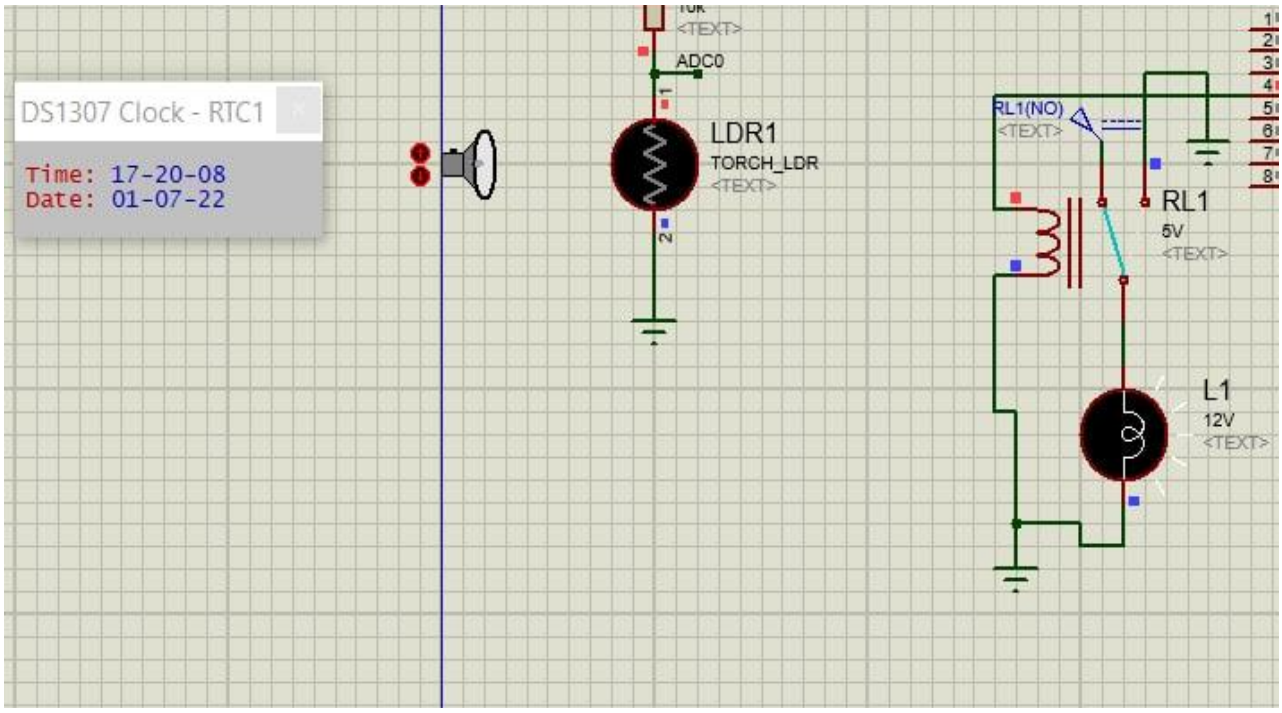


Figure 3-31 Sunset time circuit operation

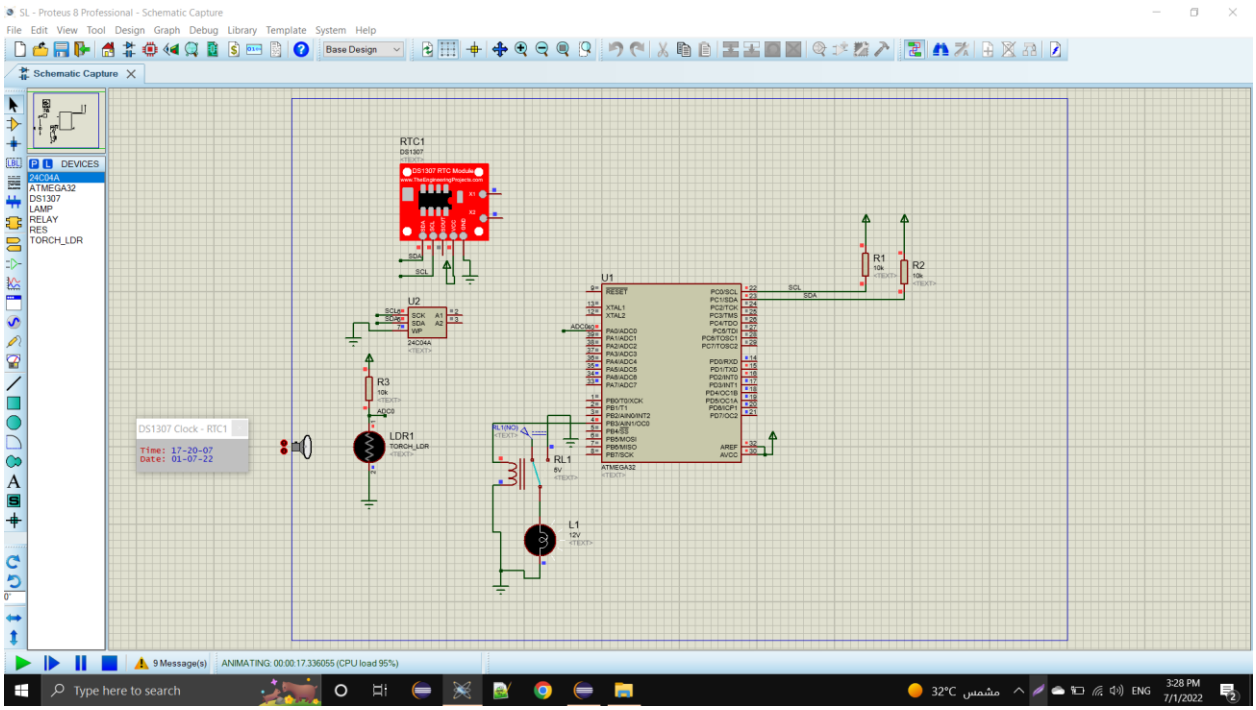


Figure 3-30 Sunset time

4. Chapter 4: Smart Parking System

4.1: Objective

- A smart parking system (SPS) is private parking for the campus area that can be used by special drivers with authorized access to the parking.
- A smart parking system solves the parking issue. A smart parking system allows the user quick access which helps in the reduction of time in searching the parking spot, reducing traffic congestion. It can be used to monitor parking systems and exhibit the parking lot situation at any given moment.

4.2: System design and analysis

4.2.1: Top-level design

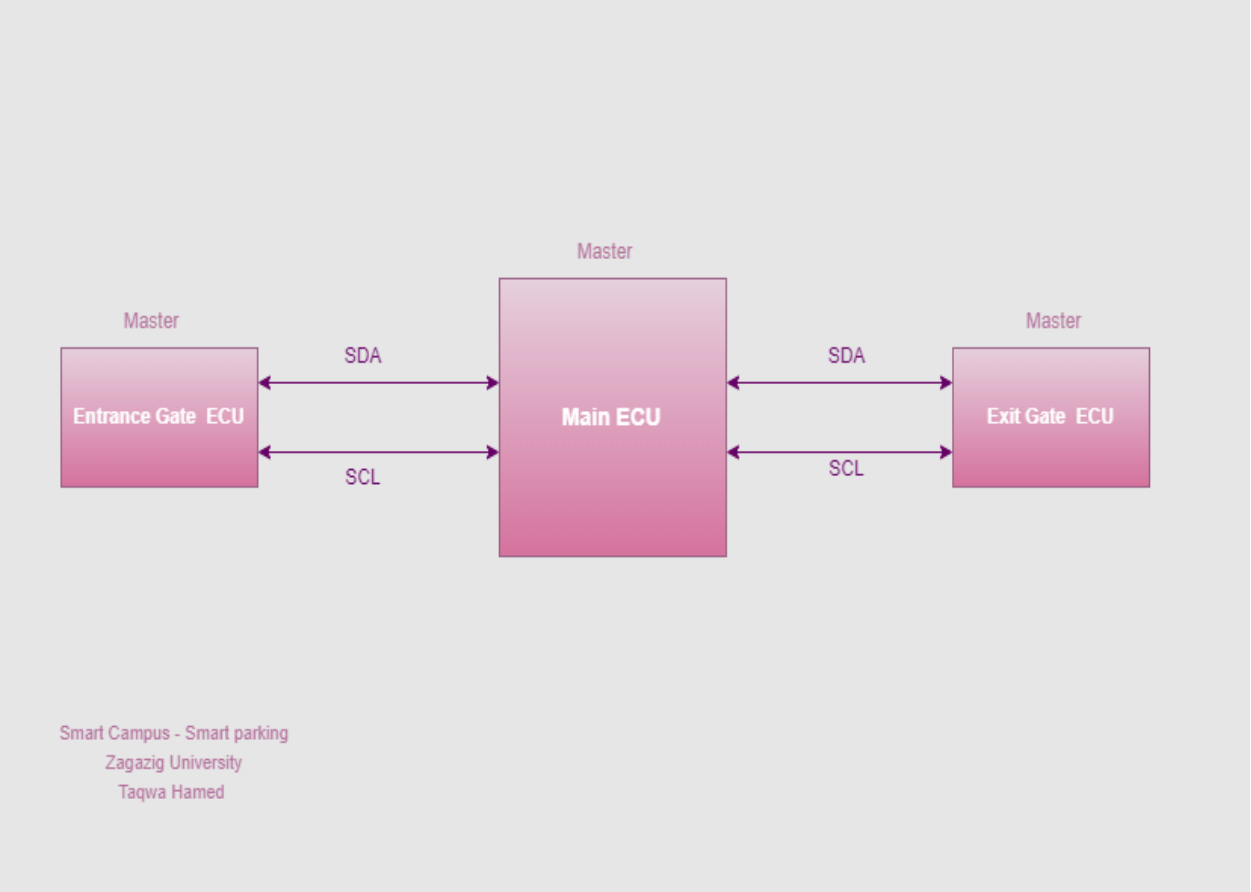


Figure 4-1 The System Architecture for Smart Parking

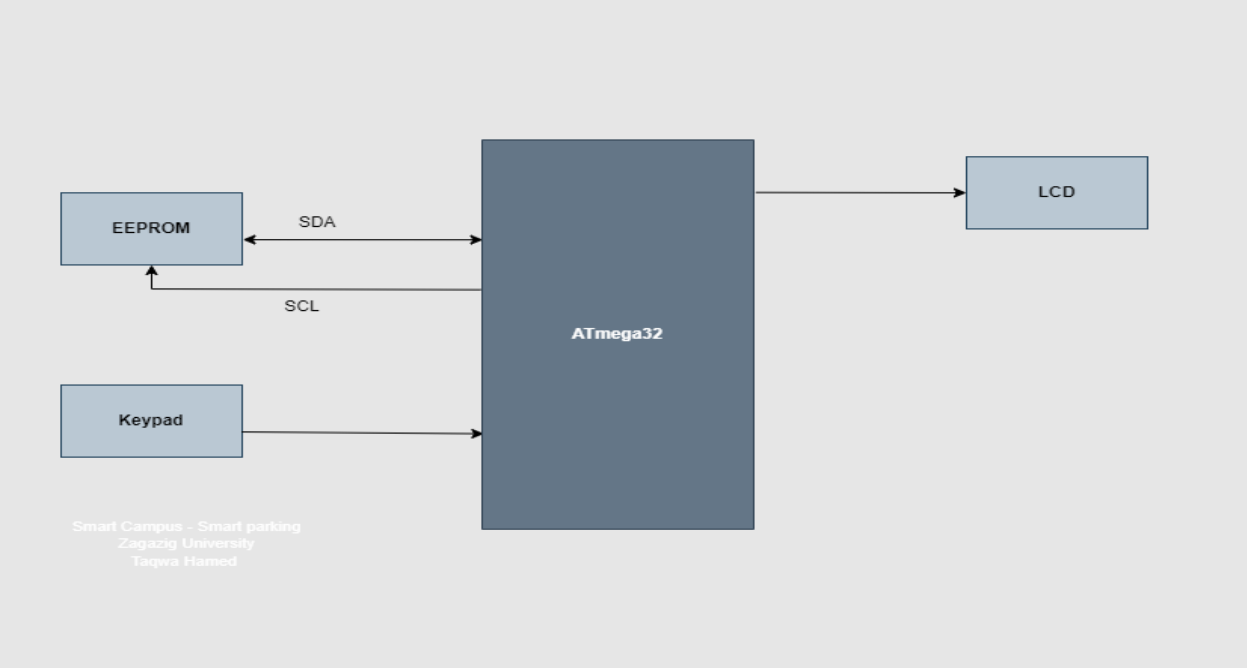


Figure 4-2 The System Architecture for the Main

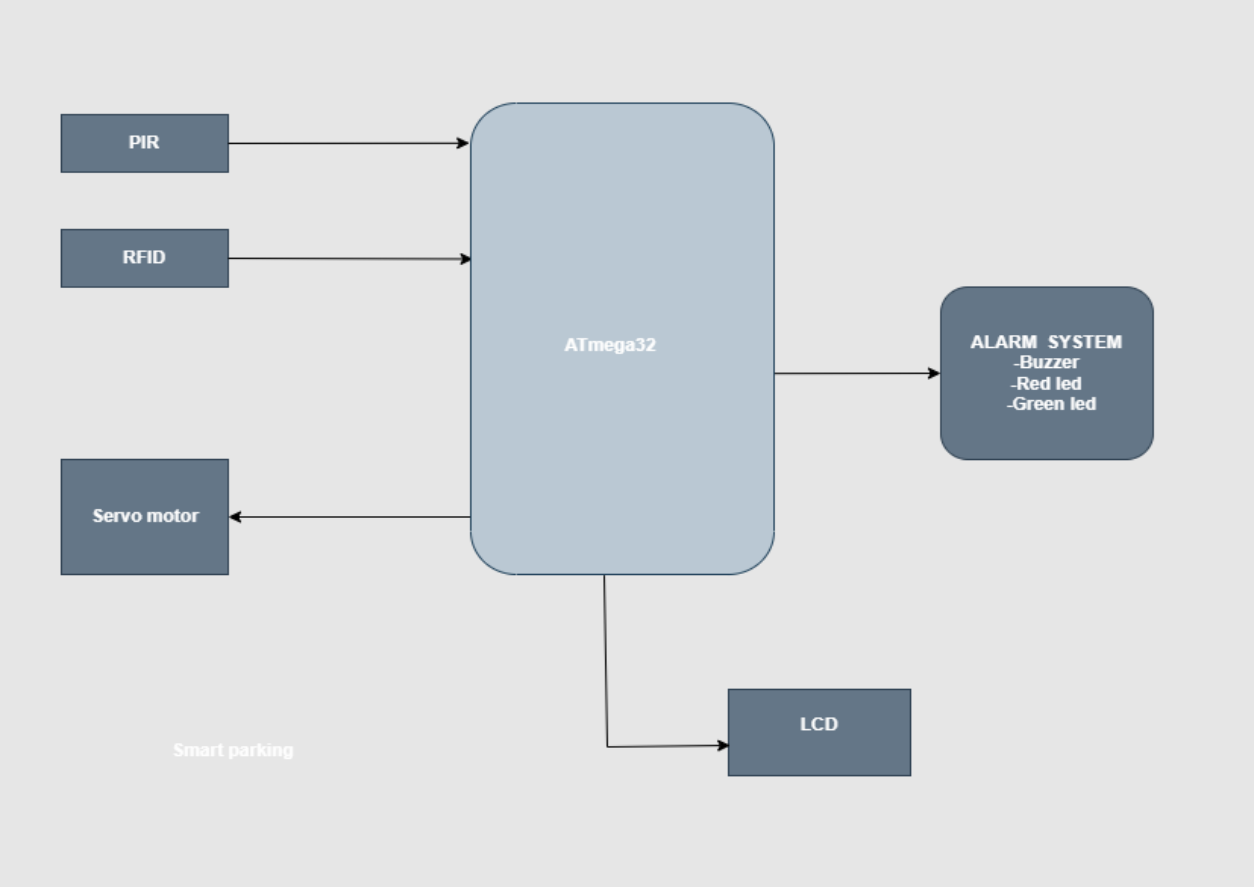


Figure 4-3 The System Architecture for Gates

4.2.2: Flow Chart

4.2.2.1: Gates

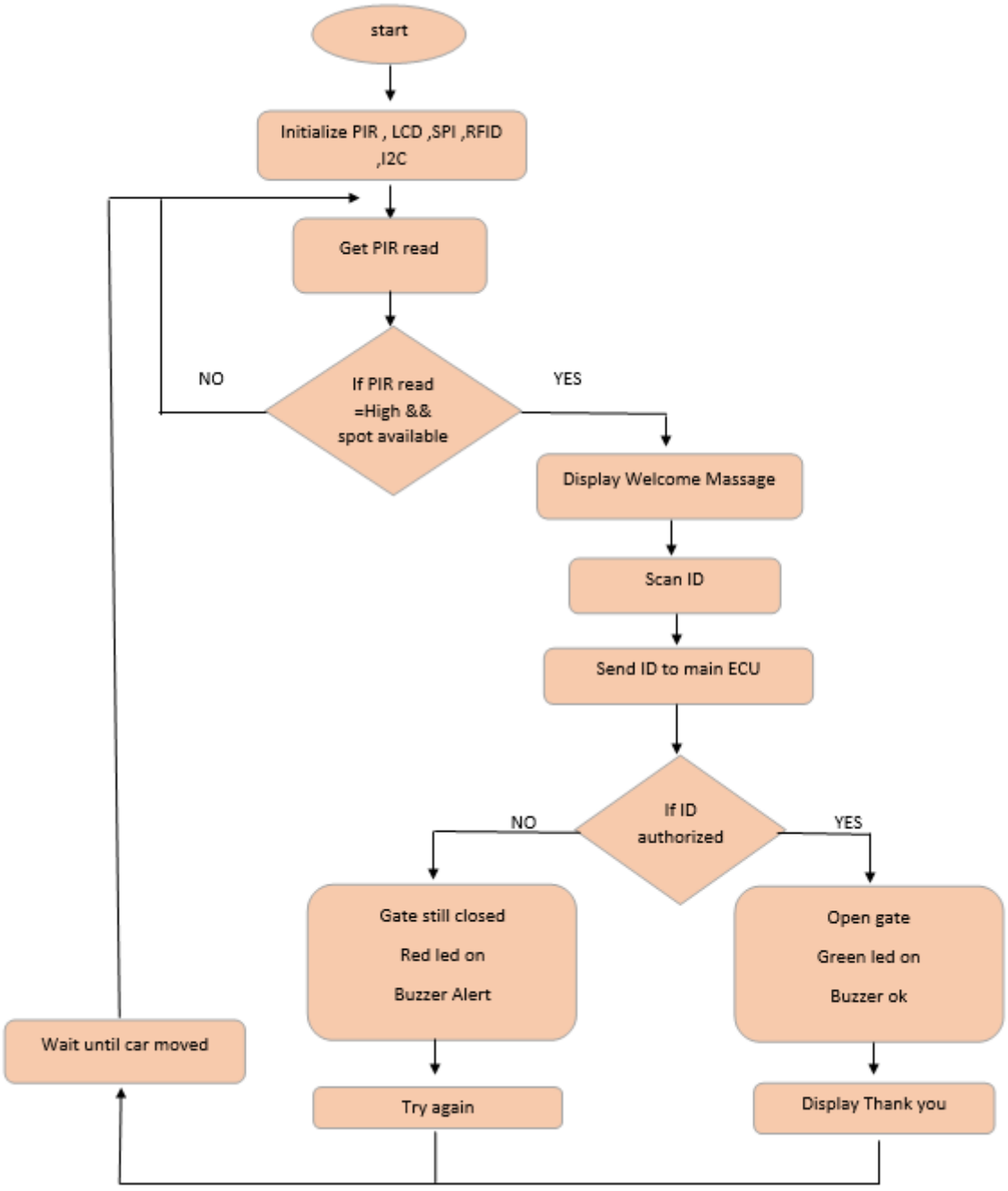
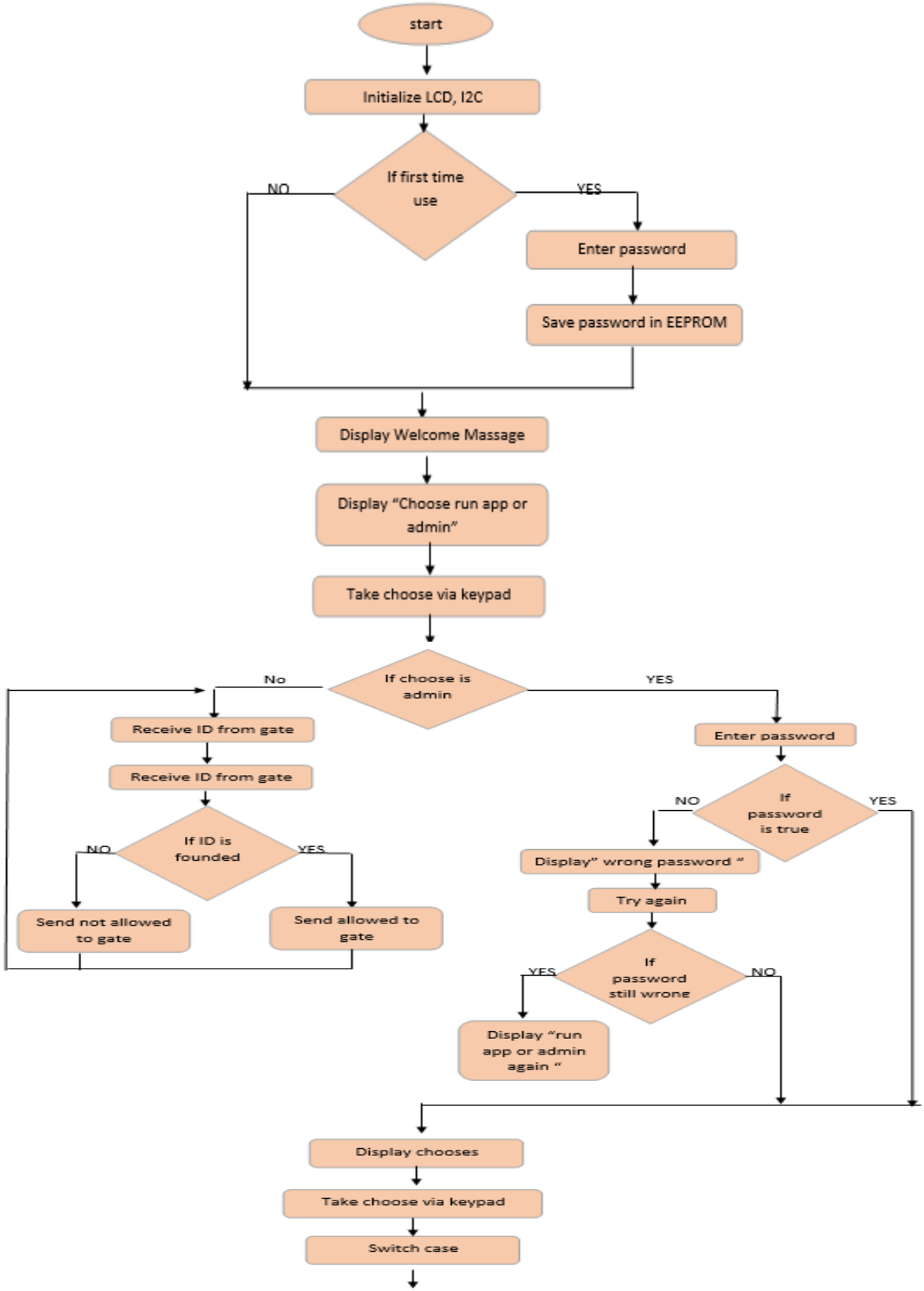


Figure 4-4 flow chart Gates

4.2.2.1: Main


start
Initialize LCD, I2C
If first time use
NO
YES
Enter password
Save password in EEPROM
Display Welcome Massage
Display "Choose run app or admin"
Take choose via keypad
If choose is admin
No
YES
Receive ID from gate
Receive ID from gate
If ID is founded
NO
YES
Send not allowed to gate
Send allowed to gate
Enter password
If password is true
NO
YES
Display" wrong password "
Try again
If password still wrone
YES
NO
Display "run app or admin again "
Display chooses
Take choose via keypad
Switch case

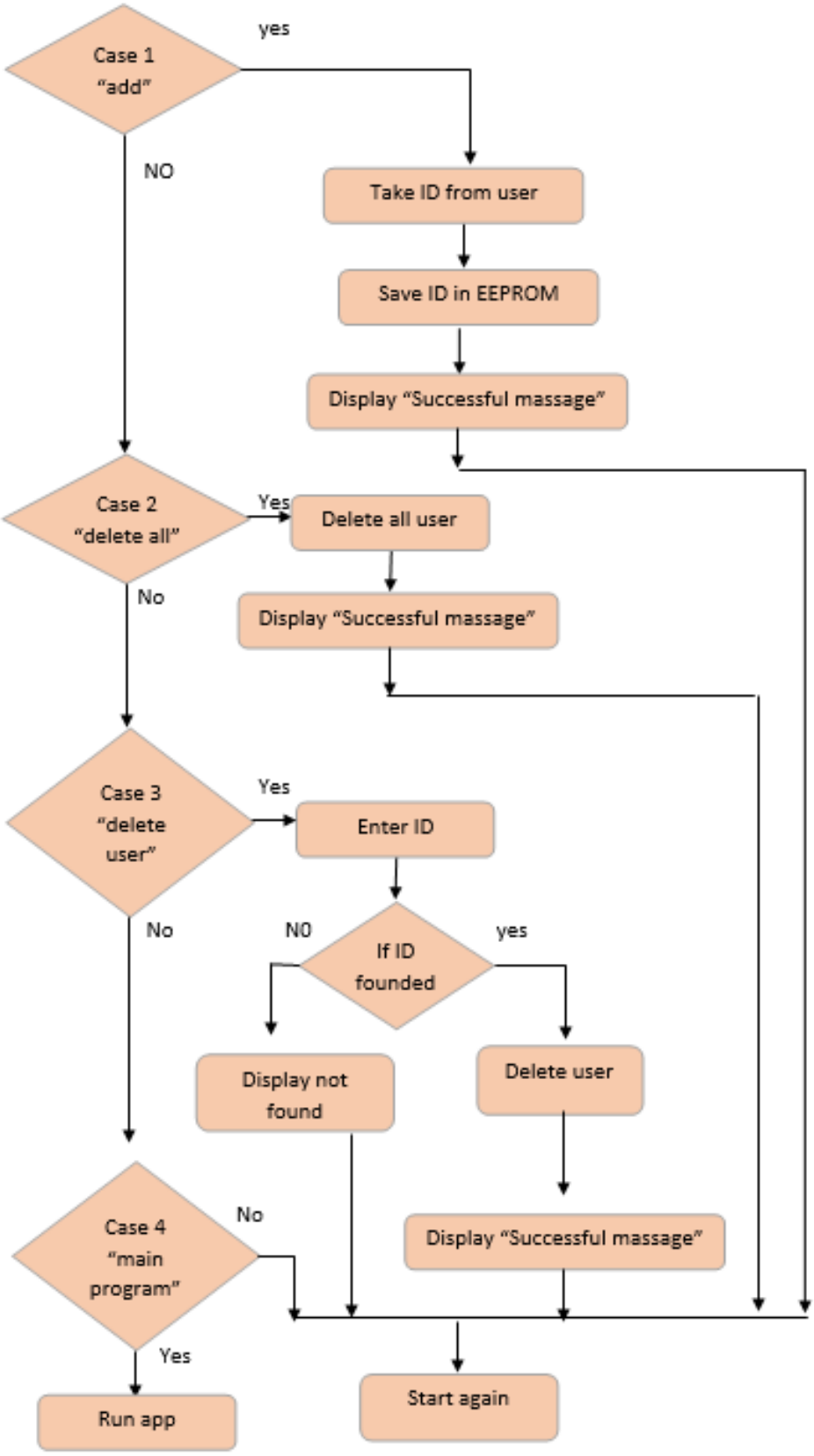


Figure 4-5 flow chart Main

4.2.3: Sequence

4.2.3.1: Main Sequence Diagram

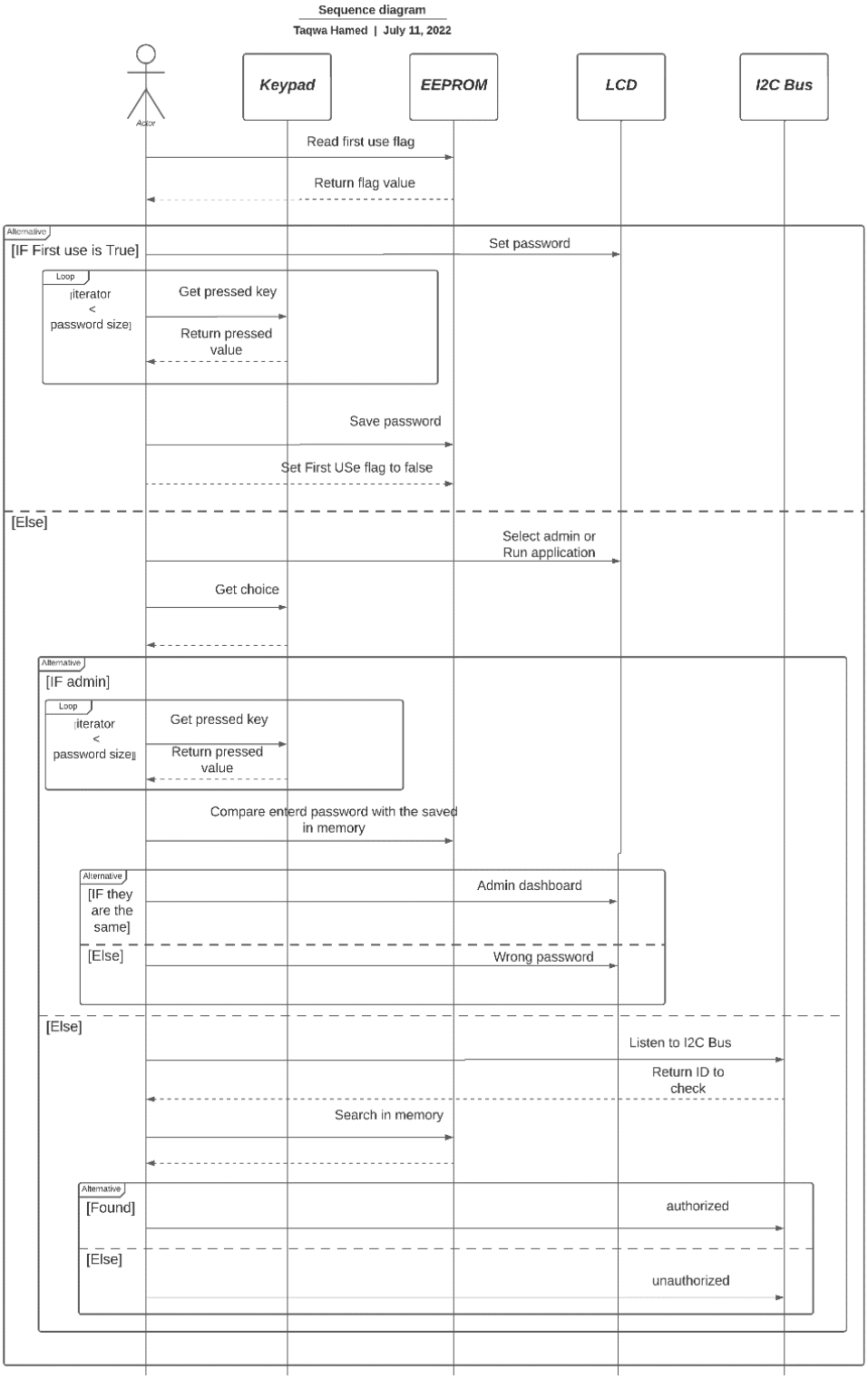


Figure 4-6 Main Sequence Diagram

4.2.3.2: Gate Sequence Diagram

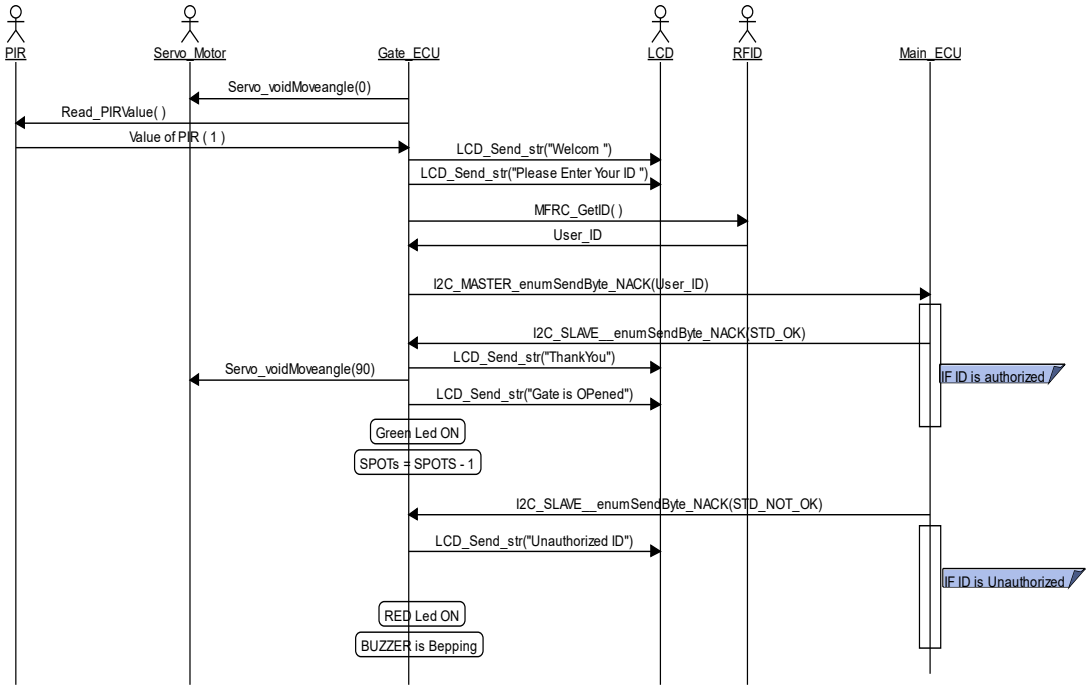


Figure 4-7 Sequence Diagram Gates

4.2.3.3: Main pseudo code

- First initialize peripherals & sensors such as LCD, Keypad, and EEPROM.
- The second step is to Read EEPROM.
  - Then check **IF** it is the first use!
    - Send message “Welcome it is your first use of the system”
    - Send message “Please set your password”

    - FOR LOOP
- Loop until reaching the max length of a password
- Save key pressed values in the array.
- Reset key pressed variable
- End Loop.

    - Write password to EEPROM
    - Change first use flag address in EEPROM
    - Send message “password was set correctly”

- **ELSE**
  - Get the password from EEPROM and save it in an array.

❖ **LOOP** for ever
- Display a message to start with Admin mode or start the app.
- Take input from the keypad
- **IF** user wanted Admin mode!
  - Loop to take the password and save it on an array.
- Then Reset key pressed variable
- Compare entered password with the password saved in EEPROM.
- **IF** the two passes are the same!
  - Login finished.
  - Start var = Admin dashboard
  - Reset key pressed variable.
- **ELSE**
  - Send message "Wrong pass try again"
  - Entering until finished three tries, then must wait 20 seconds.
- **ELSE IF**
- Start var =main program.
  - **CASE** (Admin dashboard)
  - Send message "1- ADD USER 3- MORE"
  - Send message "2- DELETE USER"
  - Take input from the keypad.
  - **IF** input = Add user
  - Start var = Add user
  - **ELSE IF** input = Delete user
  - Start var = Delete user
  - **ELSE IF** input = Second Admin Page
  - Start var = Second Admin Page
  - **ELSE**
  - Clear screen
  - Send message "Wrong choice"
  - Reset key pressed variable.

  - **CASE** (Second Admin Page)
  - Send message "4- Delete All"
  - Send message "5- Back"
  - Send message "6- main"
  - Take input from the keypad.

- **CASE** (Add User)
- Clear screen
- Send message “Enter User ID”
- Save pressed ID in array
- Convert admin string ID to number.
- **CASE** (Delete ALL)
- Clear screen
- Send message “Deleting Users”
- Call delete All users FUNC.
- Send message “Done”
- **CASE** (Delete User)
- Clear screen
- Send message “Enter User ID”
- Take input from the keypad.
- Save input in the array
- Reset key pressed variable.
- Convert admin string ID to number.
- **CASE** (Main program)
- Main ECU listens to the I2C bus for its address
- Receive the ID to check
- Listen to the I2C bus to reply
- **IF** ID is verified send ok
- **ELSE** send not ok

4.2.3.4: Gate pseudo code

Entrance Gate Pseudo Code

➢ First initialize peripherals & sensors such as LCD, I2C, PIR, SERVO, and RFID.

❖ **LOOP** for ever

- First, the gate locked (Servo angle was 0)
- **IF** PIR sense a car
- Send message “Welcome to Entrance gate”
- Send message “Enter Your ID”
- Receiving ID from RFID Card & send it to main ECU using I2C
- Receiving Flag from main ECU To check ID authorized OR not.
- **IF** ID is authorized
- Open the gate (Servo Angle=90)
- Send message” Thank you, the gate is opened”
- Green LED ON.
- **ELSE** Send message” The ID is not allowed”
- RED LED ON
- Alarm is beeping

Exit Gate Pseudo Code

- First initialize peripherals & sensors such as LCD, I2C, PIR, SERVO, and RFID.
- **LOOP** for ever
- First, the gate locked (Servo angle was 0)
- **IF** PIR sense a car
- Send message “Goodbye”
- Send message “Enter Your ID”
- Receiving ID from RFID Card & send it to main ECU using I2C
- Receiving Flag from main ECU To check ID authorized OR not.
- **IF** ID is authorized
- Open the gate (Servo Angle=90)
- Send message” Thank you, the gate is opened”
- Green LED ON.
- **ELSE** Send message” Un authorized ID”
- RED LED ON
- Alarm is beeping

4.2.4: Hardware

4.2.4.1: SPI

- Serial peripheral interface:
- The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between two or more devices. the data is frame-less, and throughput is very high.
- The **ATmega32 SPI** includes the following features:
  - Full-duplex, Three-wire Synchronous Data Transfer
  - Single Master / Multi Slave
  - LSB First or MSB First Data Transfer
  - Seven Programmable Bit Rates
  - End of Transmission Interrupt Flag
  - Write Collision Flag Protection
  - Wake-up from Idle Mode
  - Double Speed (CK/2) Master SPI Mode

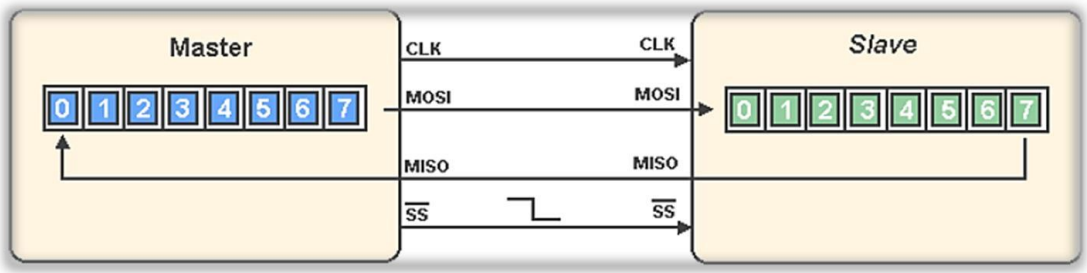


Figure 4-8 SPI master/slave interface

- **Working principle:**

- SPI consists of two shift registers, one on the master and the other on the slave side. Also, there is a clock generator on the master side that generates the clock for the shift registers.
- The Master pulls low the slave select (SS) line to the desired Slave to initiate the communication cycle.
- The Master starts to generate the required clock pulses on the SCK line to the slave and Master.
- To transfer data, The data is shifted between Master and Slave.
- Data is shifted From Master to Slave on the MOSI line (Master output / Slave input), and from Slave to Master on the MISO line (Master input / Slave output).
- After data is translated, The Master will pull high the Slave Select (SS) line.

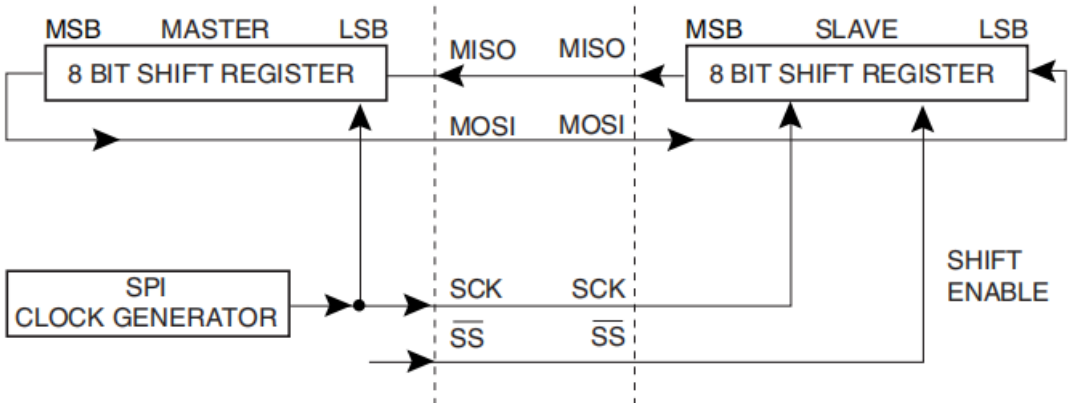


Figure 4-9 SPI master/slaver interconnection

4.2.4.2: I2C

- As found in Chapter Three (smart lightning) in sec two (Hardware toles) Number 5(3.2.5.5: I2C)

4.2.4.3: PWM

- As found in Chapter Three (smart lightning) in sec two (Hardware toles) Number 4 (3.2.5.4: PWM)

4.2.4.4: RFID

Figure 4-10 RFID

- Radio Frequency Identification (RFID) is the wireless non-contact use of radio frequency waves.
- RFIDs use an electromagnetic field to identify objects or track the objects automatically.
- RFID systems usually consist of RFID readers, RFID tags, and antennas.
- RFID tags need not be in line with the sensor unit. It can be in any random position.
- There are two types of RFID readers fixed readers and mobile readers.
- The RFID reader is a network-connected device that can be portable or permanently attached.
- RIFD tags need a power source or battery to power up the tags circuit.
- Tags that have a stronger power source also have a longer read range.
- RFID data can be updated in real-time.
- 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.
- **Every RFID system consists of:**
  - ✓ Scanning antenna
  - ✓ Transceiver
  - ✓ transponder
- **RFID reader mainly consists of:**
  - ✓ RF signal generator
  - ✓ Microcontroller
  - ✓ Receiver/signal detector

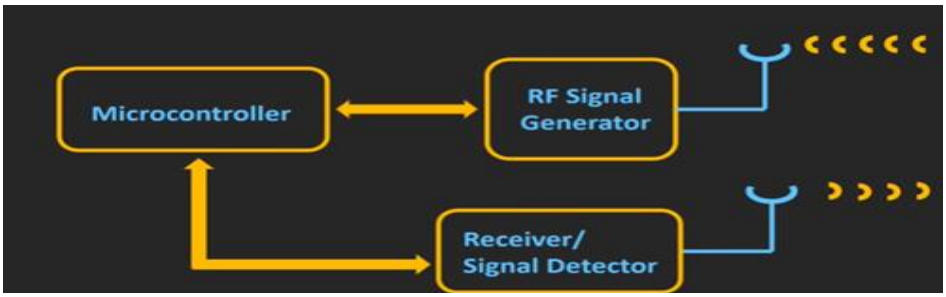


Figure 4-11 RFID reader Components

- **RFID tag basic components:**
  - ✓ Transponder
  - ✓ Rectifier circuit
  - ✓ Controller
  - ✓ Memory

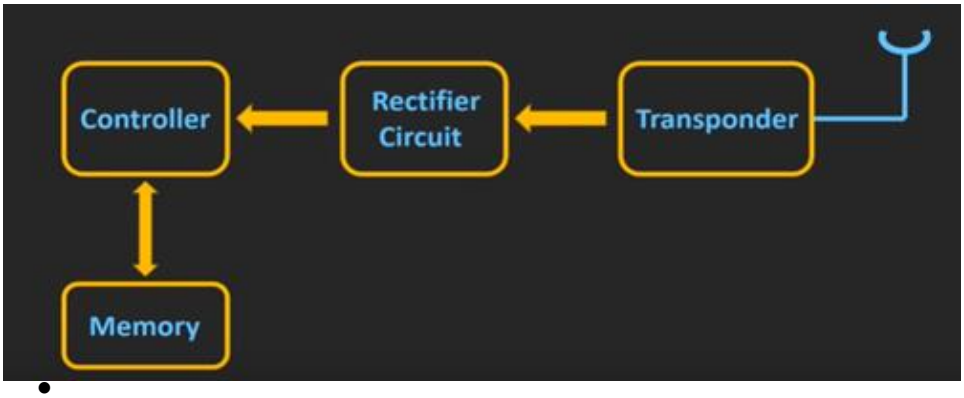


Figure 4-12 Figure 4-13 RFID Tag Components

- **RFID tags are three types:**
  1- Active tag:
  It has its power supply but transmits a signal back to the reader also they depend on its power supply.
  2- Passive tag:
  It is not having a power supply. So, this passive tag depends on the radio waves which are coming from the RFID reader from the source of energy.
  3- Semi-passive tag:
  It has its power supply but for transmitting feedback signal back to the RFID reader they used to depend on the signal which is coming from the RFID reader.
- **Frequency of RFID used for operations:**
  1- Low frequency:
    a. Operation range 125: 134 kHz.
    b. Travel ranges up to 10 cm.
  2- High frequency:
    a. Operation range13.56 MHZ.
    b. Travel ranges up to 1 m.
  3- Ultra-high frequency:
    a. Operation range 860: 960 MHZ.
    b. Travel ranges up to 10: 15 m.

- **RFID work:**
  1- RFID tags contain an integrated circuit and an antenna, which are used to transmit data to the RFID reader when the reader requests the data from the tag.
  2- The reader then converts the radio waves to a more usable form of data.
  3- Information collected from the tags is then transferred through a communications interface to a host computer system, where the data can be stored in a database and analyzed later.
  4- A Reader consists of a Radio Frequency module and an antenna that generates a high-frequency electromagnetic field.
  5- On the other hand, the tag is usually a passive device, meaning it doesn't contain a battery. Instead, it contains a microchip that stores and processes information, and an antenna to receive and transmit a signal.
  6- To read the information encoded on a tag, it is placed near the Reader (does not need to be within a direct line of sight of the reader).
  7- The reader generates an electromagnetic field that causes electrons to move through the tag's antenna and subsequently power the chip.
  8- The powered chip inside the tag then responds by sending its stored information back to the reader in the form of another radio signal this is called backscatter.
  9- The backscatter is detected and interpreted by the reader which then sends the data out to a microcontroller.
- **RFID working principle in different frequencies:**
  - **Low frequency and high frequency:**
    (Based on the inductive coupling (near field coupling))
    1- RFID reader continuously sends radio waves at a particular frequency. The radio waves which is sent by the reader are for three purposes.
       - ✓ It includes enough power to tag
       - ✓ It provides synchronization clock tag to passive tag
       - ✓ Acts as a carrier for return data from the tag
    2- The field which is generated by the RFID reader is used to couple with the antenna of an RFID tag.
    3- The mutual coupling voltage will get induced across the coil of the RFID tag. Some portion of voltage is rectified and uses a power supply for the controller and memory elements.
    4- RFID reader sends radio waves of a particular frequency so that voltage will be induced across the coil at a particular frequency.
    5- This induced voltage is used to synchronize the clock of the controller.
    6- If we connect the load across the coil, the current will start flowing through the connected load. If we change the impedance of the

load, simultaneously the current flowing through the load will also change (Ohms Law).

7- If we switch ON/OFF load current will also be ON/OFF. The switching of current and the rate of change of current also generate the voltage in an RFID reader. The switching ON/OFF of the load is known as load modulation.

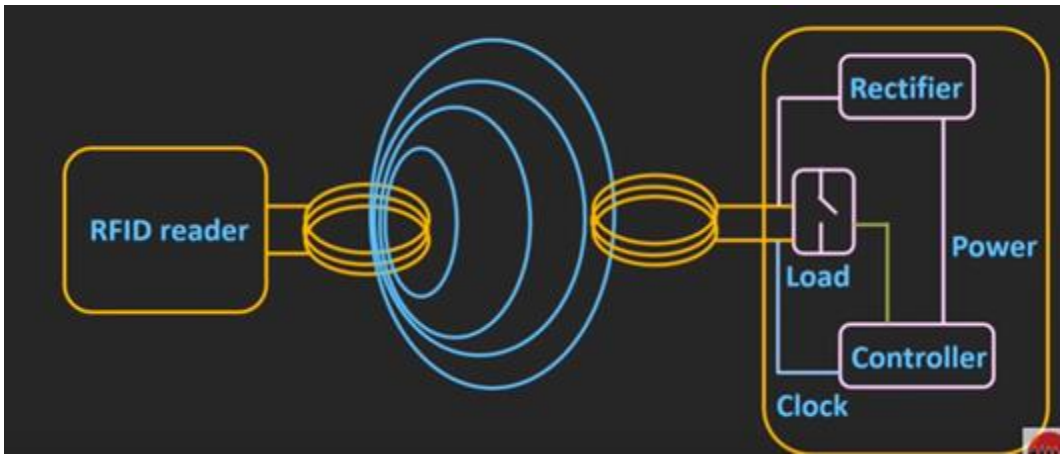


Figure 4-14 working principles of Low frequency and high frequency

- **Ultra-high frequency (far-field coupling):**
  (Based on the electromagnetic coupling (far-field coupling))
  1- RFID reader continuously sends the waves at a particular frequency toward the tag.
  2- In response, the tag is sending a weak signal to the RFID reader.
  3- This week signal will be sent back to an RFID reader which is known as a backscattered signal.
  4- The Intensity of the backscattered signal is depending upon the load matching across the coil by changing the condition of the load we can change the intensity of the backscattered signal.
  5- If we can change the condition of data that is stored in the RFID tag, then data can be sent back to the RFID reader.
  6- In far-field coupling, RFID and tag are few so the initial stage signal sent by the reader should be strong.

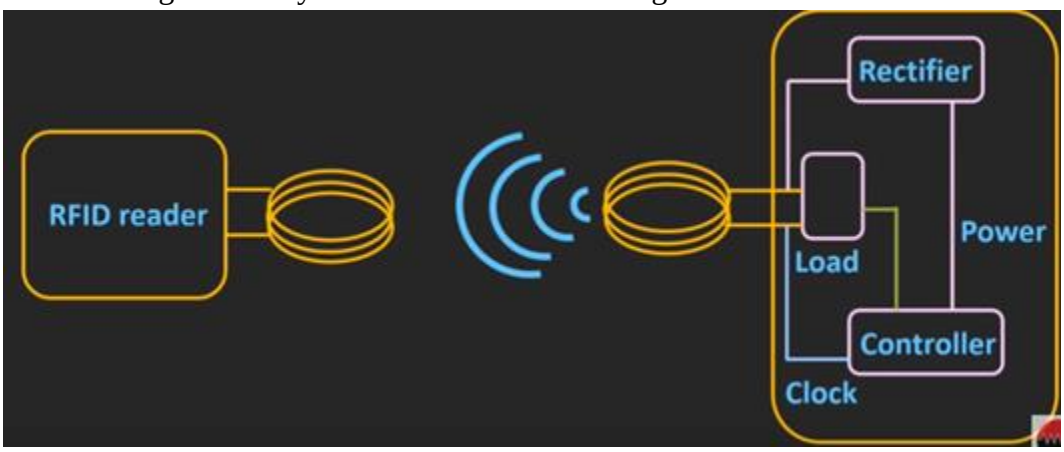


Figure 4-15 working principle of Ultra-high frequency

- **Applications:**
  - Used in institutions like libraries, hospitals, etc.
  - Transportation and logistics.
  - Access control.
  - Sports
  - Animals tracking

4.2.4.5: EEPROM

Figure 4-16 EEPROM

- EEPROM is electrically erasable programmable read-only memory.
- EEPROM is a nonvolatile memory.
- Atmega32 has an internal 1024-byte(1Kbyte) EEPROM.
- Any byte within an EEPROM may be erased and rewritten.
- Write/Erase Cycles(endurance): 100000 for EEPROM which is internal inside atmega32.
- Fast to read not to write.
- Data retention: 20 years at 85°C/100 years at 25°C
- The primary tradeoff for this improved functionality is a higher cost to store small amounts of data.

- EEPROM IC communicates with help of the I2C protocol to interface it with an MCU just power the IC (usually with 3.3V or 5V) and connect the communication lines.
- **It has an 8-pin DIP package.**
  - ✓ Pin 1,2,3 user configured chip select pins, useful during cascading
  - ✓ Pin 4 is connected to the ground of the circuit.
  - ✓ pin 5 serial data pin for I2C communication.
  - ✓ Pin 6 serial clock pin for I2C communication.
  - ✓ Pin 7 is connected to Vss write is enabled, if connected to Vcc write is disabled.
  - ✓ Pin 8 for connecting to power

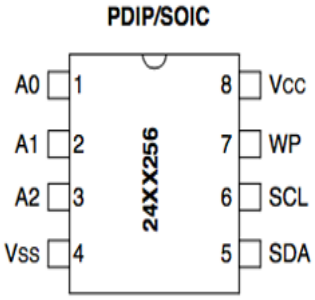


Figure 4-17 EEPROM

4.2.4.6: Servo Motor

Figure 4-18 Servo Motor

- A servo motor is an electronic circuit for controlling the direction of rotation, as well as the position, of the motor shaft.
- There are two types of servo motors AC and DC servo motors.
- A servo motor has three wires: two are designated for power supply (Vcc and Ground) and the third wire is for the control signal.
- The control input to a servo is a pulse width modulated (PWM) signal, generally of frequency 50 Hz.
- The pulse width values for different angular positions of this servo are provided.

Pulse width (ms)	Angular position (° CCW)
0.7	0 (min)
1.1	45
1.5	90
1.9	135
2.3	180 (max)

Figure 4-19 pulse width values for different angular

4.2.4.7: Keypad

Figure 4-20 Keypad

- A keypad is a set of mechanical switches that are arranged in a Matrix to minimize the number of used pins.
- 4x4 keypad consists of 4 rows and 4 columns.
- Used as an input device to read the key pressed by the user.

4.2.4.8: 16×2 LCD

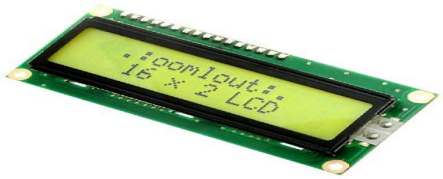


Figure 4-21 LCD

- It is a type of flat panel display which uses liquid crystals in its primary form of operation. mostly depend on multi-segment LEDs.
- Used to display information.

4.3: Description

- **The system consists of three main sub-systems (3 ECUs)**

  1- ECU1: Admin dashboard control

  2- ECU2: Entrance-gate control

  3- ECU3: Exit-gate control

4.3.1: Admin Dashboard Control

- there are two options to enter the parking as an Admin or user.

- **First**, to enter the parking as an admin you have three tries to enter your password if you used them, you would wait 20 seconds to enter more tries. Then you will be able to add and delete users from their IDs.

Figure 4-22 Admin Dashboard Control

1- Add User

- Adding a user by entering the user's id
- The user's id will be saved in EEPROM
- If the user enters a used ID, we will display 'found ID'

2- Delete User

- Deleting user with entering user's id
- If ID is not found in EEPROM, we will display 'ID not found'

3- Delete all users

- If the admin selects the option of deleting all users, All users will be deleted from EEPROM

- **Second**, run the app option if the user enters the id the first step is to check if the id is saved in EEPROM or not if it is saved the user will be able to enter the gate easily.

4.3.2: Entrance-Gate Control

- At the entrance gate, the LCD display "Welcome in Entrance gate" and then " Please Enter your ID".
- Scan ID from the user RFID reader.
- Send ID to main ECU then Search ID on EEPROM after search Return ID
  In one bit YES (authorized) or NO (unauthorized).
- In the case that the ID is **authorized**:
  - The gate will be opened, and a green led on and displayed on the LCD "Thank you".
  - The number of empty spaces in the garage is reduced by one.
  - In case the garage is complete with the number of cars, a message appears that there is no place.
- In the case that the ID is **unauthorized:**
  - Displayed on LCD is "Unauthorized ID".
  - Red led on and the alarm is beeping.

4.3.3: Exit-Gate Control

- At the exit gate, the LCD displays "Goodbye" and then " Please Enter your ID".
- Scan ID from the user RFID reader
- Send ID to main ECU then Search ID on EEPROM after search Return ID
  In one bit YES (authorized) or NO (unauthorized)

- In the case that the ID is **authorized:**
  - The gate will be opened, and a green led on and displayed on the LCD “Thank you”.
  - The number of empty spaces in the garage is increased by one.
  - LCD displays "garage has space"
- In the case that the ID is **unauthorized:**
  - Displayed on LCD is “Unauthorized ID”.
  - Red led on and the alarm is beeping.

4.4: Simulation Results

4.4.1: Gate Test

- a) No cars
- Condition: PIR doesn't detect any objects (PIR-output =0)
- Expected: not enter the if condition and just display messages on LCD

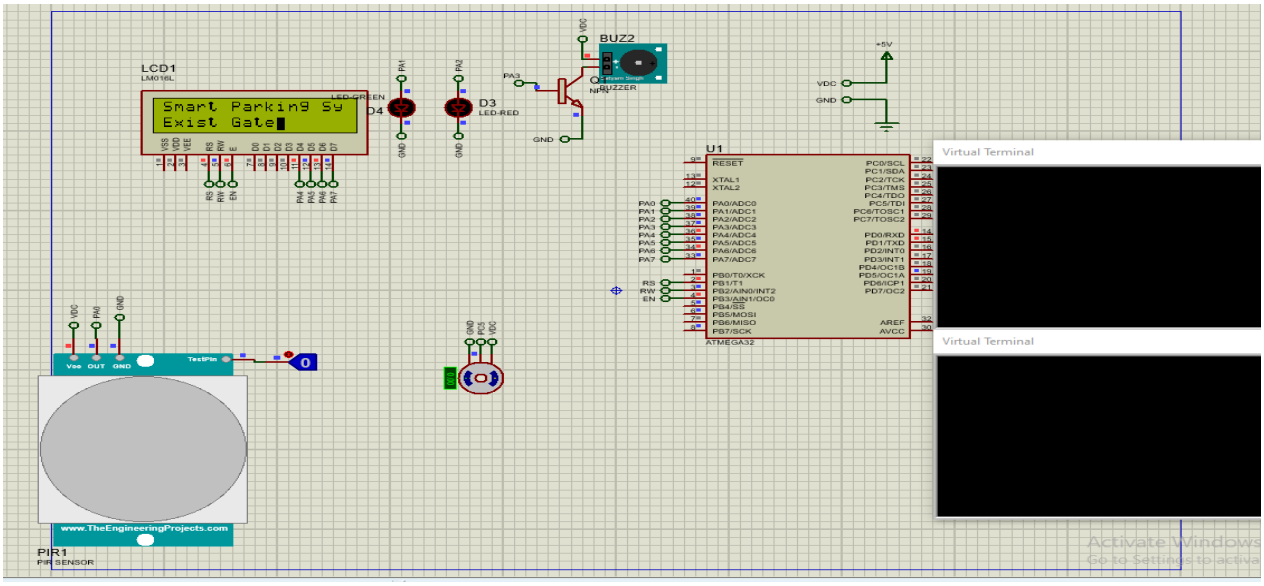


Figure 4-23 In case No car

- b) there's a car
- Condition: PIR detects a car (pir_output = 1)
- Expected: enter the if condition and wait to scan id_tag then dissidents the status of id if it authorized or unauthorized

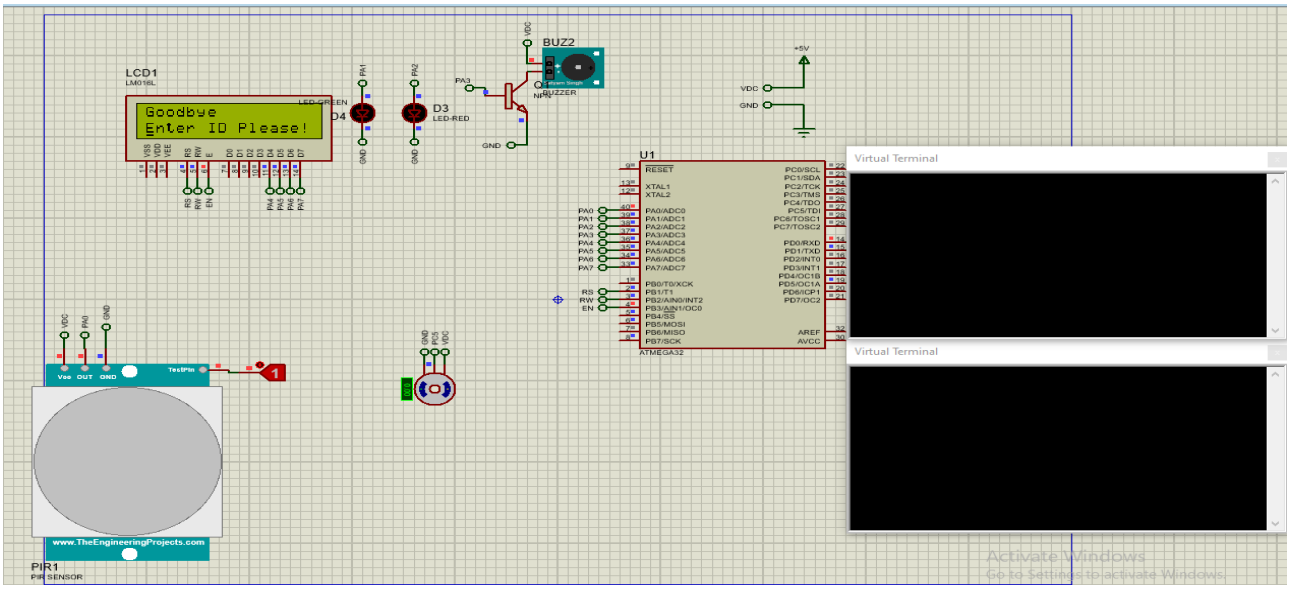


Figure 4-24 In case there's a car

- b-1) authorized id
- Condition: stored in a database (EEPROM) the main ECU sends a command that tells the gate it's authorized.
- Expected: the gate will be opened green LED will be turned on wait until the car passed then the gate will be closed again.

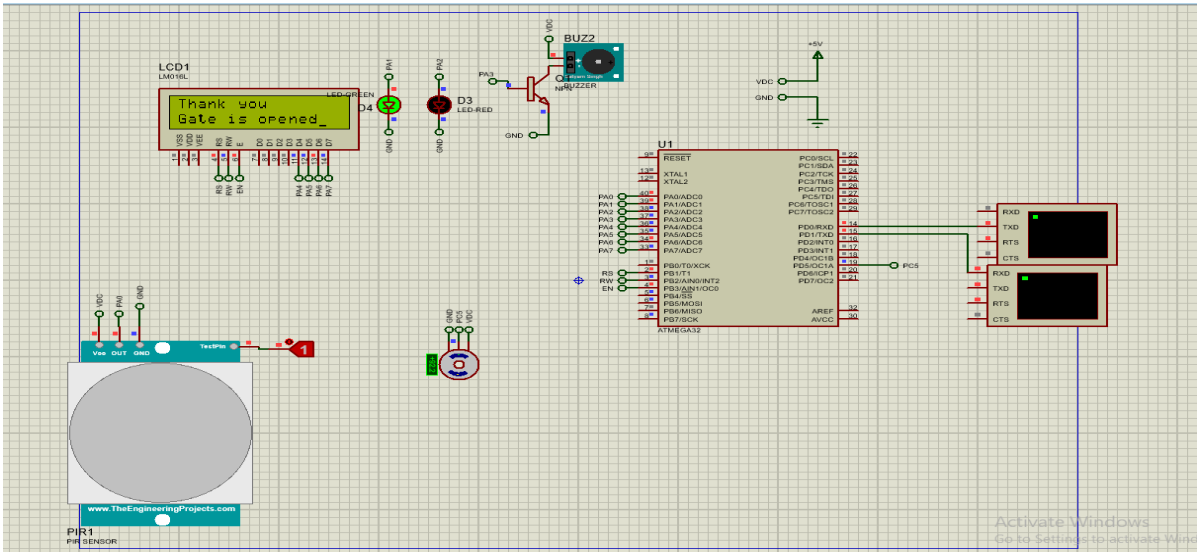


Figure 4-25 In case authorized id

- b-2) unauthorized id
- Condition: id is not stored in a database (EEPROM) the main ECU sends a command that tells the gate it's unauthorized.
- Expected: gate still being closed; the red LED will be turned on and Buzzer is beeping.

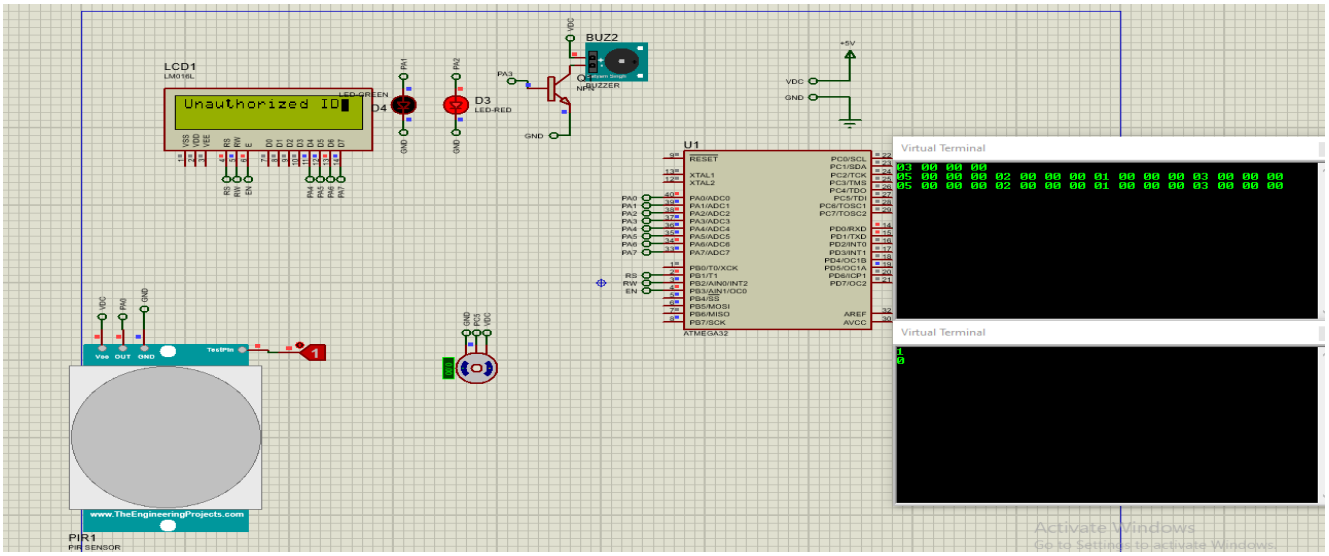


Figure 4-26 unauthorized id

4.4.2: Admin control Test

- When the App first runs the HMI(LCD) prompts the admin to set a password for the system which will be used to give access to the admin to control the system.
- Password will be saved to the Non-volatile EEPROM

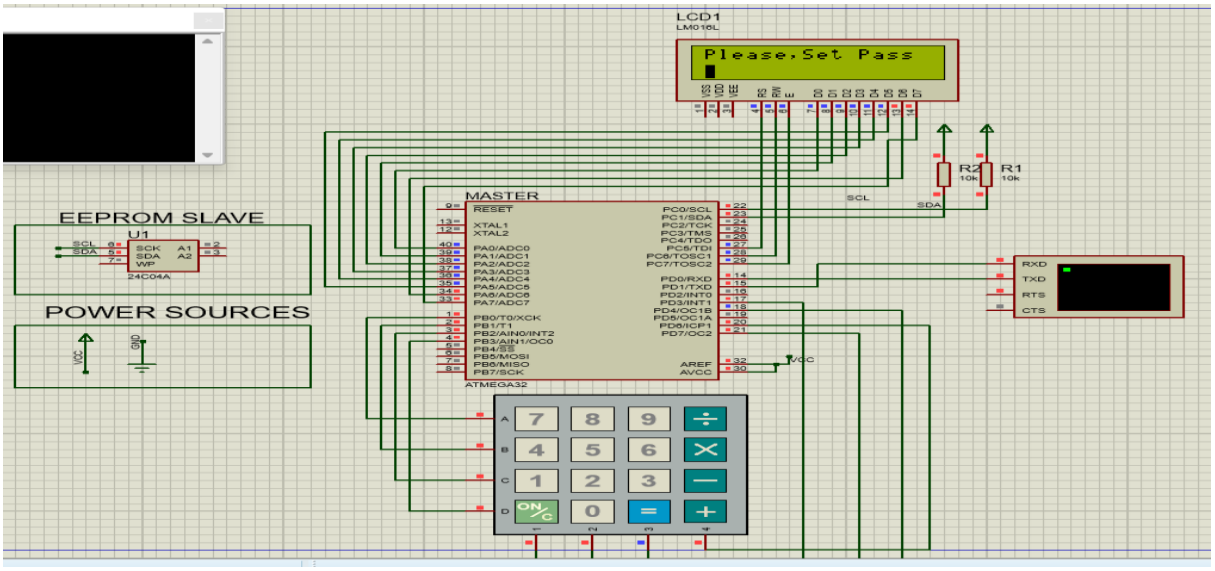


Figure 4-27 Admin control test

- HMI will keep prompting the admin to ask for Running the System or Run as Admin to control the system.

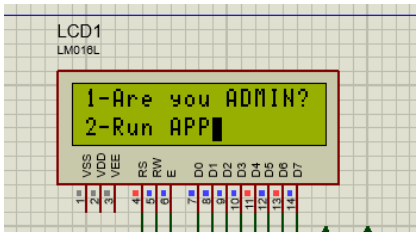


Figure 4-28 asks for running system or run as admin

- Test cases for admin
  - Wrong password

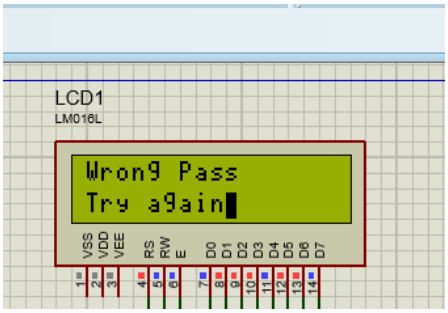


Figure 4-29 In case Wrong password

  - 3 times wrong password

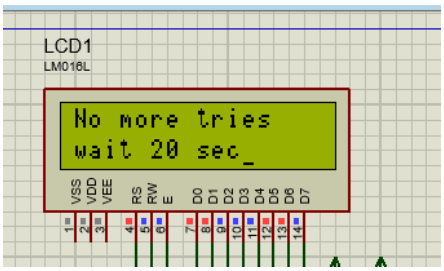
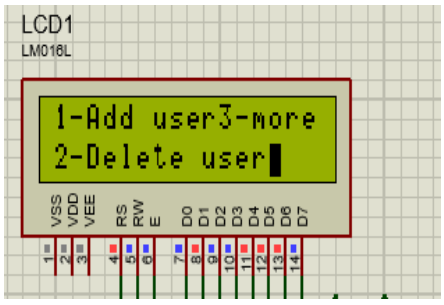


Figure 4-30 In case trying 3 times wrong password

  - After Login with a verified password
    - Admin now can
      1- Add user
      2- Delete user



- The second Mode is Running the App
    - ➢ It just handles the gate requests and checks if ID has access or not.

2.5: Smart Parking Circuit

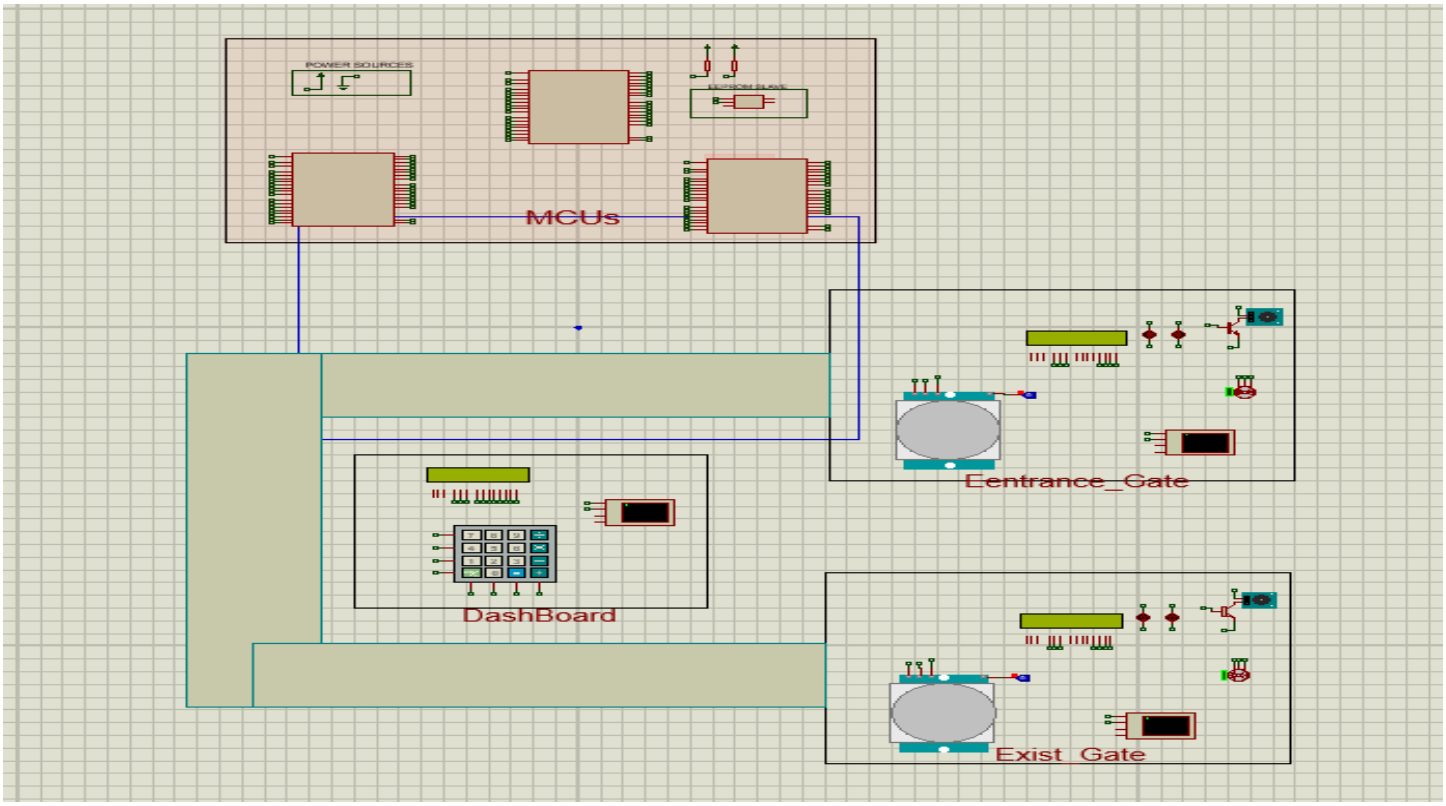


Figure 4-31 smart parking circuit

5. Chapter 5: Part 2 Necessary Background

1. Necessary Background

1.1. Machine Learning

The typical method for getting a computer to perform useful tasks is to have a human programmer create rules—a computer program—that must be followed to transform input data into the right answers, much like Lady Lovelace did when she created the Analytical Engine's step-by-step instructions. Machine-learning reverses this: The device appears to examine the incoming data and the accompanying responses to determine the rules. ought to be (see figure 5.1). Instead of being explicitly programmed, a machine learning system is educated. programmed. It is given many examples pertinent to a task, and it recognizes statistical patterns in these examples that eventually allow the system to develop rules. to automate the process. If you wanted to automate the process of tagging your vacation photos, for instance, you might give a machine learning system a large sample of photos that have already been tagged by people, and the system would learn statistical rules for matching certain photos to specific tags.

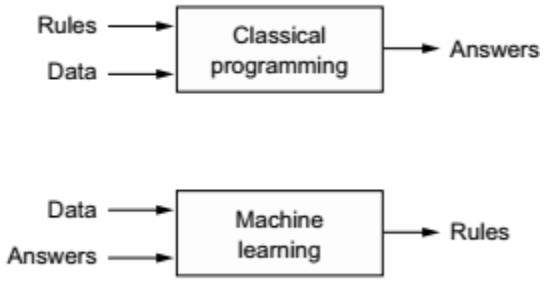


Figure 5-1 Machine learning a new programming paradigm.

1.2. Deep Learning

Deep learning is a particular branch of machine learning that emphasizes the learning of successive layers of increasingly meaningful representations. Deep learning is a novel approach to learning representations from data. The "deep" in "deep learning" refers to multiple layers of representations rather than any form of deeper understanding that can be attained by the approach. The depth of the model refers to the number of layers that go into a data model. Other names for the area could have been layered representations learning or hierarchical representations learning. Today's deep learning techniques frequently comprise tens or even hundreds of representational layers that are learned automatically through exposure to training data. Other machine learning techniques, commonly referred to as shallow learning, are more inclined to concentrate on learning just one or two layers of data representations (for example, taking a pixel histogram and then applying a classification algorithm).

1.3. **Artificial Neural Network (ANN)**

As you can see, ANNs are made up of a large number of neurons that are organized into layers to carry out computations and forecast results. It is also possible to refer to this design as a multilayer perceptron, which is more logical because it suggests that the network is made up of multiple-layered perceptrons. This neural network architecture is referred to by the acronyms MLP and ANN interchangeably. Each node in the MLP diagram shown in figure 2.2 is referred to as a neuron. Let's focus on the perceptron, the most fundamental part of the neural network, before moving on to how MLP networks function. Understanding how several perceptrons work together to learn data features will be easier after you have a basic understanding of how a single perceptron operates.

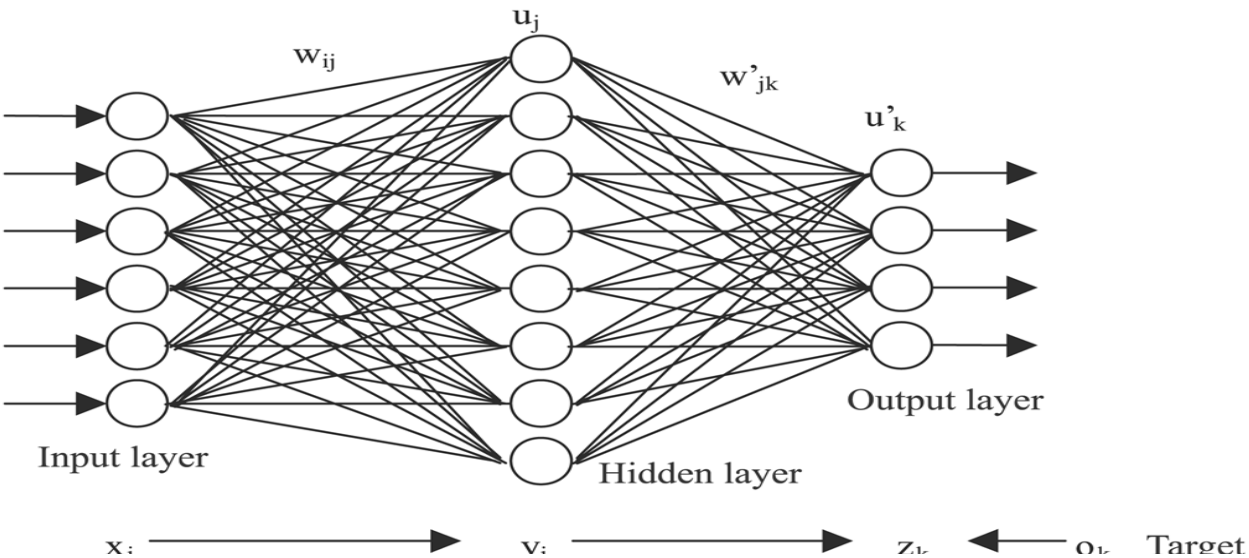


Figure 5-2 An artificial neural network consists of layers of nodes, or neurons connected with edges.

1.4. **What is a perceptron?**

The perceptron, which consists of just one neuron, is the simplest type of neural network. The perceptron conceptually works similarly to a biological neuron (figure 2.3). An electrical signal is received by a biological neuron from its dendrites, modulated in different ways, and then fired through its synapses only when the sum of the input signals is stronger than a predetermined threshold. A subsequent neuron receives the output, and so on. The artificial neuron carries out the following two sequential tasks to mimic the behavior of biological neurons: It determines whether to fire the output 1 if the signal exceeds a specific threshold or 0 if the signal does not exceed the threshold by computing the weighted sum of the inputs to represent the entire intensity of the input signals. Not all input features are equally valuable or significant, as we covered in chapter 1. Each input node is given a weight value, known as its connection weight, to symbolize this and indicate its significance.

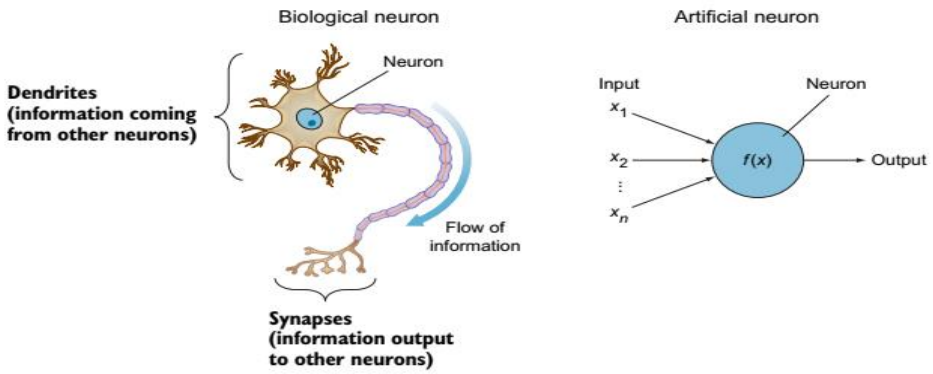


Figure 5-3 Artificial neurons were inspired by biological neurons. Different neurons are connected to each other by synapses that carry information.

nput vector: The feature vector that is fed to the neuron. It is usually denoted with an uppercase X to represent a vector of inputs (x1, x2, . . ., XN).

Weights vector: Each x1 is assigned a weight value w1 that represents its important to distinguish between different input data points.

Neuron functions: The calculations performed within the neuron to modulate **the input signals**: the weighted sum and step activation function. ϒ *Output*—Controlled by the type of activation function you choose for your network. For a step function, the output is either 0 or 1. Other activation functions produce probability output or float numbers. The output node represents the perceptron prediction.

1.5.*How does the perceptron learn?*

Trial and error are how the perceptron learns from its errors. It adjusts the weights' values up and down like knobs until the network is trained.

- **The perceptron's learning logic goes like this:**
  The neuron calculates the weighted sum and applies the activation function to make a prediction $\hat{y}$. This is called the feedforward process:
  $\hat{y}$ = activation $(\sum xi \cdot wi + b)$
- It compares the output prediction with the correct label to calculate the error:
  error = $y - \hat{y}$
- It then updates the weight. If the prediction is too high, it adjusts the weight to make a lower prediction the next time and vice versa.
- Repeat!
  This process is repeated many times, and the neuron continues to update the weights to improve its predictions until step 2 produces a very small error (close to zero), which means the neuron's prediction is very close to the correct value. At this point, we can stop the training and save the weight values that yielded the best results to apply to future cases where the outcome is unknown.

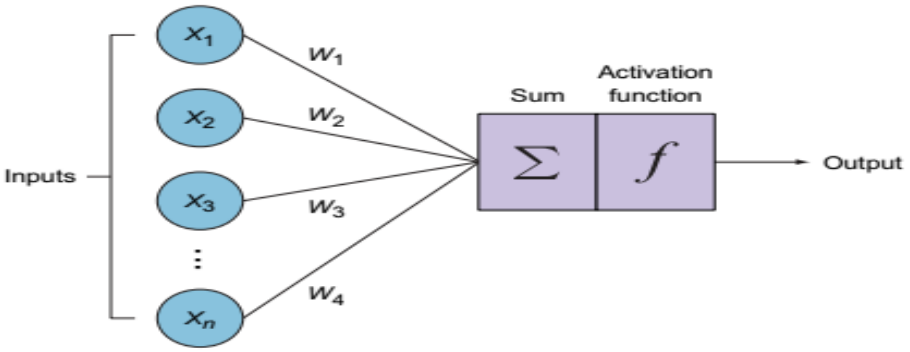


Figure 5-4 Input vectors are fed to the neuron, with weights assigned to represent importance.

1.6. *Understanding how deep learning works*

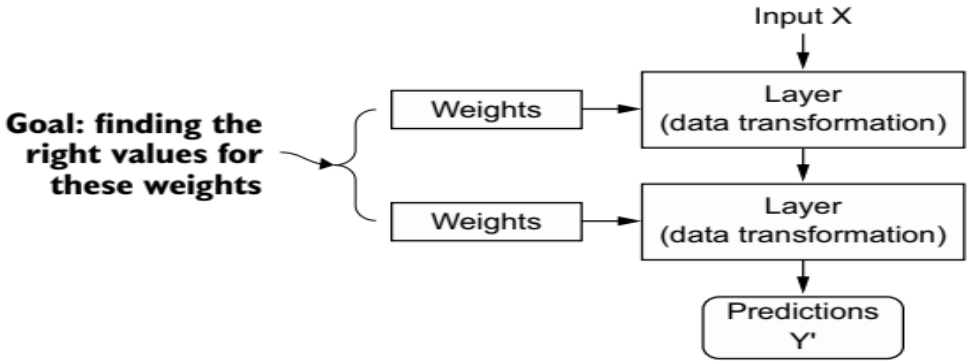


Figure 5-5 A neural network is parameterized by its weights.

That mapping inputs (like photos) to targets (like the label "cat") is what machine learning is all about, and that it is accomplished by looking at several examples of input and target. Additionally, you are aware that deep neural networks use a lengthy series of straightforward data transformations (layers) to perform this input-to-target mapping and that these data transformations are acquired through exposure to instances. Let's now examine the specifics of how this learning occurs. The weights of a layer, which are essentially a collection of numbers, contain the definition of what a layer does to its incoming data. Technically speaking, we would state that a layer's weights parameterize the change that layer does. (Weights are also frequently referred to as a layer's parameters.) Finding a set of weight values for each layer in a network such that it can accurately map example inputs to their corresponding targets is what is meant by "learning" in this context. A deep neural network, however, can store tens of millions of parameters. Given that changing the value of one parameter may change how the others behave, finding the right settings for each one may seem like an impossible effort.

You must first be able to perceive something to control it. You must be able to gauge how far a neural network's output deviates from your expectations if you want to be able to regulate it. This is the responsibility of the network's loss function, which is also referred to as the objective function or cost function. The loss function computes a distance score, which measures how well

the network performed in this particular case, using the network predictions and the true goal (what you wanted the network to output).

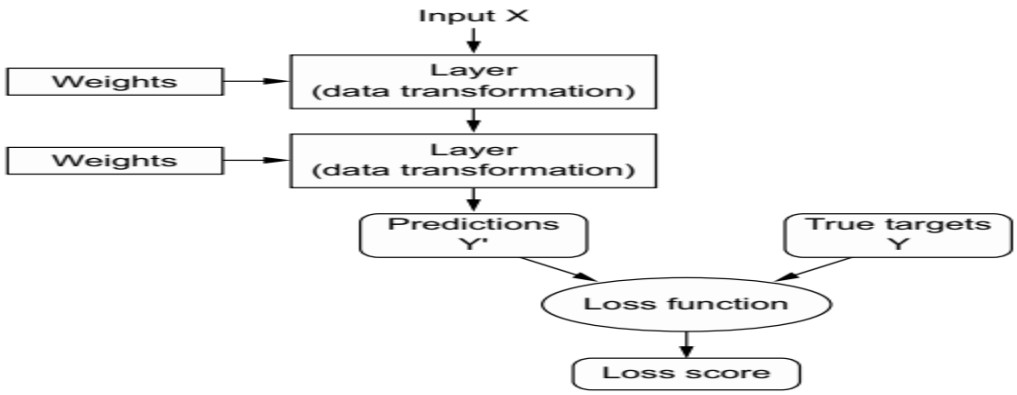


Figure 5-6 A loss function measures the quality of the network's output.

The key to deep learning is to utilize this score as a feedback signal to slightly change the weights' values in a way that reduces the current example's loss score. The optimizer, which implements the Backpropagation algorithm, the key algorithm in deep learning, is in charge of making this modification.

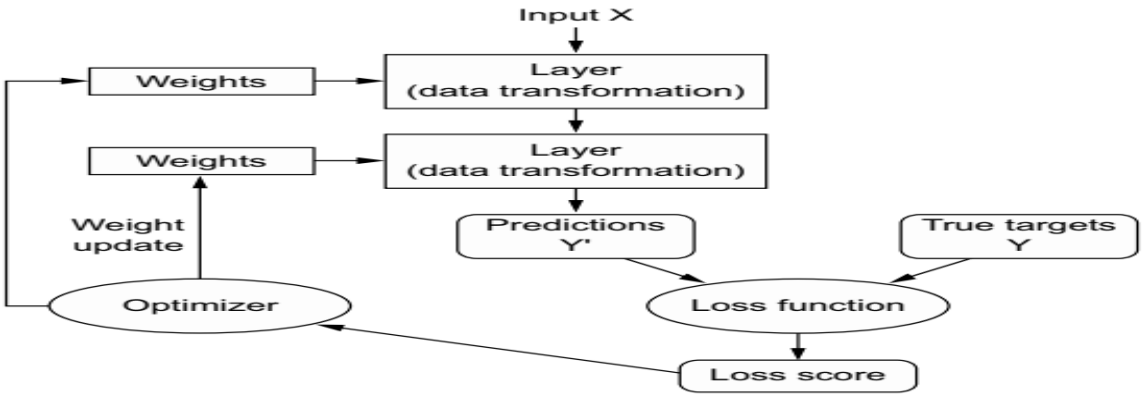


Figure 5-7 The loss score is used as a feedback signal to adjust the weights.

The network simply performs a series of arbitrary changes because the weights of the network are initially given random values. Its productivity is obviously considerably below what it should be, and as a result, the loss score is extremely high. However, the weights are somewhat changed in the right direction with each example of the network processes, resulting in a lower loss score. The training loop produces weight values that minimize the loss function when it is run a sufficient number of times (usually tens of iterations over thousands of samples). A network that has a small loss has outputs that are as close as possible to the targets: this is a trained network.

1.7. **Convolutional neural networks (CNN) Architecture**

CNN function similarly to regular neural networks in that they include numerous layers that enable each layer to find increasingly complicated characteristics. Edges and lines are learned in the first layer of convolutions, followed by features that are a bit more complicated (circles, squares, etc.), features that are much more complex (such as facial features, automobile wheels, dog whiskers, and so on), and so on. In hidden layers, neurons are stacked on top of one another; weights are initiated randomly and learned during network training; finally, activation functions are applied, error (y - y) is calculated, and the error is backpropagated to update the weights. The method is the same. The distinction is that for the feature learning component, we use convolutional layers as opposed to conventional fully connected layers.

1.7.1. **The high-level architecture of CNNs**

- Input layer
- Convolutional layers for feature extraction
- Fully connected layers for classification
- Output prediction

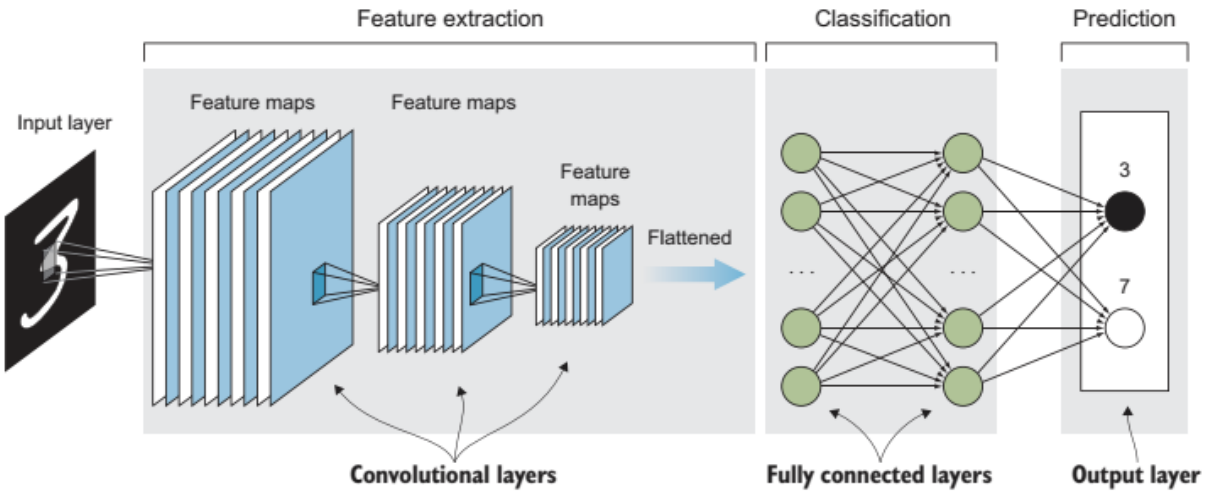


Figure 5-8 The CNN architecture consists of the following: input layer, convolutional layers, fully connected layers, and output prediction.

- Feed the raw image to the convolutional layers.
- The image passes through the CNN layers to detect patterns and extract features called *feature maps*. The output of this step is then flattened to a vector of the learned features of the image. Notice that the image dimensions shrink after each layer, and the number of feature maps (the layer depth) increases until we have a long array of small features in the last layer of the feature-extraction part. Conceptually, you can think of this step as the neural network learning to represent more abstract features of the original image.

- The flattened feature vector is fed to the fully connected layers to classify the extracted features of the image.
- The neural network fires the node that represents the correct prediction of the image. Note that in this example, we are classifying two classes (3 and 7). Thus the output layer will have two nodes: one to represent digit 3, and one for digit 7.

1.7.2. Basic components of a CNN

There are three main types of layers that you will see in almost every convolutional network

1. Convolutional layer (CONV)
2. Pooling layer (POOL)
3. Fully connected layer (FC)

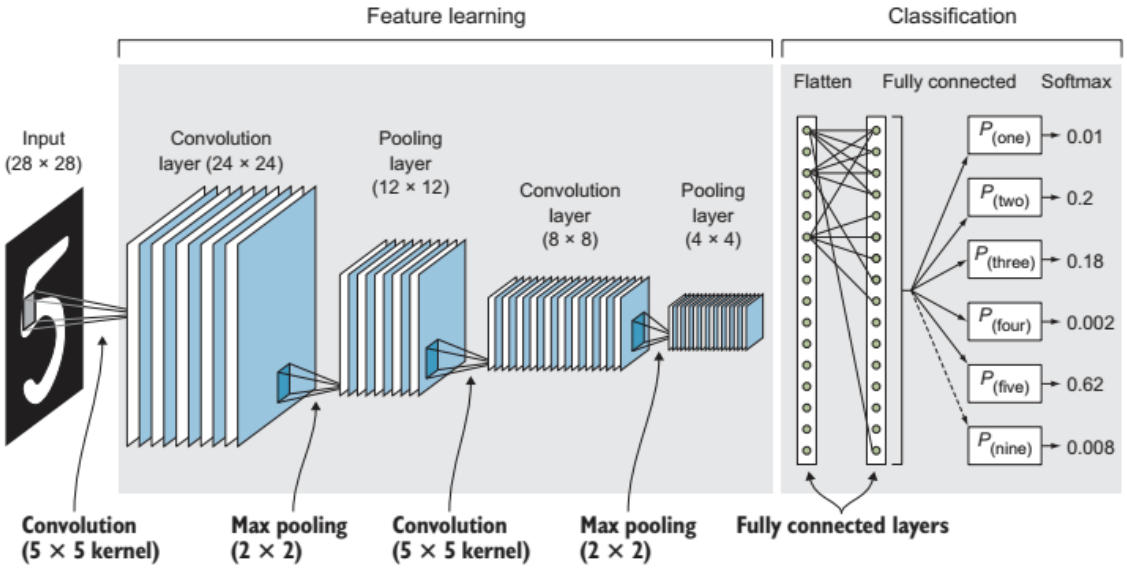


Figure 5-9 The basic components of convolutional networks are convolutional layers and pooling layers to perform feature extraction and fully connected layers for classification.

1.7.3. *Convolutional layers*

Convolution in mathematics is the process of two functions to get a third altered function. The input picture serves as the first function in the context of CNNs, and the convolutional filter serves as the second. To create a changed image with new pixel values, we will apply several mathematical procedures.

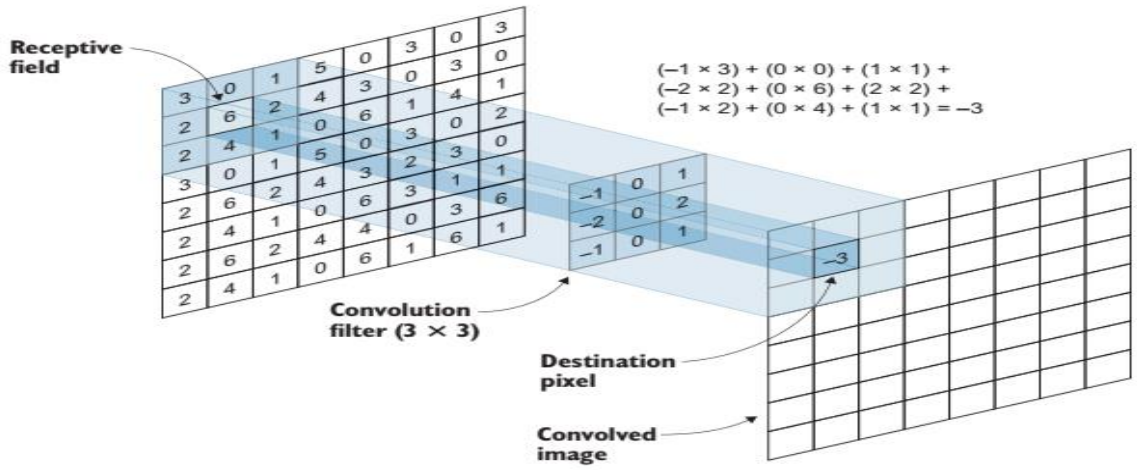


Figure 5-10 A 3 × 3 convolutional filter is sliding over the input image.

Keeping this diagram in mind, here are some facts about convolution filters:

- The small 3 × 3 matrix in the middle is the convolution filter, also called a kernel.
- The kernel slides over the original image pixel by pixel and does some math calculations to get the values of the new "convolved" image on the next layer.

What are the kernel values?

In CNN, the convolution matrix is the weights. This means they are also randomly initialized and the values are learned by the network (so you will not have to worry about assigning its values).

- **CONVOLUTIONAL OPERATIONS**
  - The math should look familiar from our discussion of MLPs. Remember how we multiplied the input by the weights and summed them all together to get the weighted sum?
  - weighted sum = $x1 \cdot w1 + x2 \cdot w2 + x3 \cdot w3 + ... + xn \cdot wn + b$
  - We do the same thing here, except that in CNNs, the neurons and weights are structured in a matrix shape. So we multiply each pixel in the receptive field by the corresponding pixel in the convolution filter and sum them all together to get the value of
  - the center pixel in the new image.

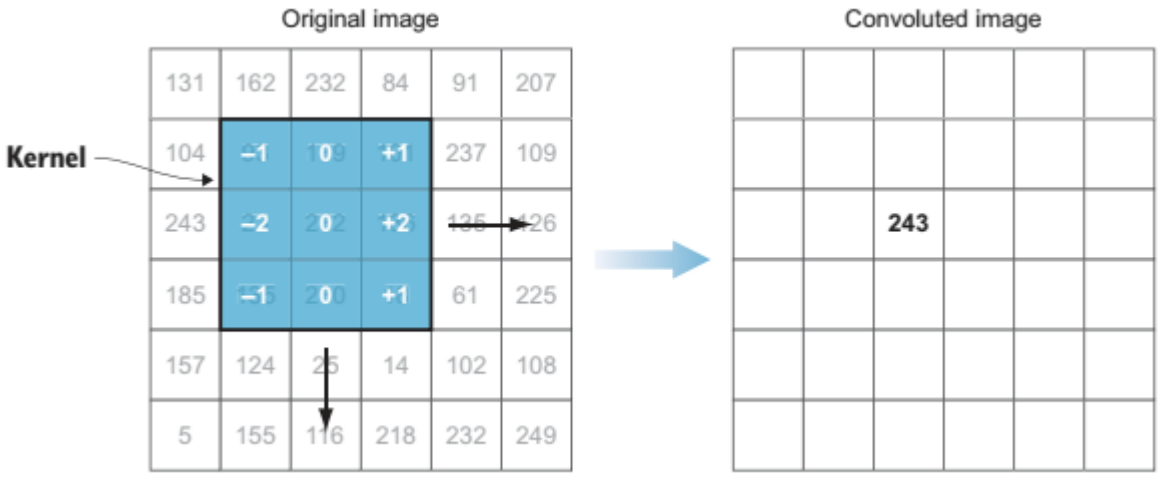


Figure 5-11 Multiplying each pixel in the receptive field by the corresponding pixel in the convolution filter and summing them gives the value of the center pixel in the new image.

(93 × –1) + (139 × 0) + (101 × 1) + (26 × –2) + (252 × 0) + (196 × 2) + (135 × –1) +(240 × 0) + (48 × 1) = 243 The filter (or kernel) slides over the whole image. Each time, we multiply every corresponding pixel element-wise and then add them all together to create a new image with new pixel values. This convolved image is called a *feature map* or *activation map*.

```
from keras.layers import Conv2D
model.add(Conv2D(filters=16, kernel_size=2, strides='1', padding='same',
activation='relu'))
```

One line of code creates the convolutional layer. We will see where this line fits in the full code later in this chapter. Let's stay focused on the convolutional layer. As you can see from the code, the convolutional layer takes five main arguments. it is recommended that we use the ReLU activation function in the neural networks' hidden layers. That's one argument out of the way. Now, let's explain the remaining four hyperparameters that control the size and depth of the output volume:

- **Filters**: the number of convolutional filters in each layer. This represents the depth of its output.
- **Kernel size**: the size of the convolutional filter matrix. Sizes vary: 2 × 2, 3 × 3, 5 × 5.
- **Stride**.
- **Padding.**

CNN, the convolutional layers are the hidden layers. And to increase the number of neurons in hidden layers, we increase the number of kernels in convolutional layers. Each kernel unit is considered a neuron. For example, if we have a 3 × 3 kernel in the convolutional layer, this means we have 9 hidden units in this layer. When we add another 3 × 3 kernels, we have 18 hidden units. Add another one, and we have 27, and so on. So, by increasing the number of kernels in a convolutional layer, we increase the number of hidden units, which makes our network more complex and able to detect more complex patterns. Figure 5.12 provides a representation of the CNN layers that shows the number-of-kernels idea.

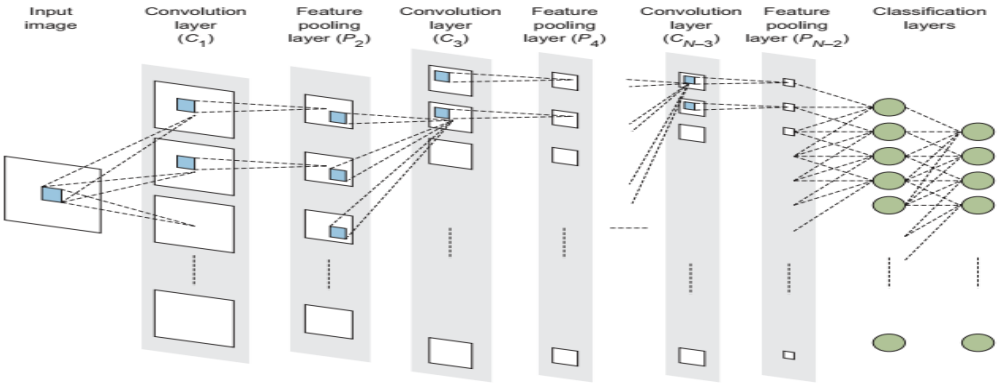


Figure 5-12 Representation of the CNN layers that shows the number-of-kernels idea.

- **KERNEL SIZE:**
  - A convolution filter is also known as a *kernel.* It is a matrix of weights that slides over the image to extract features. The kernel size refers to the dimensions of the convolution filter. kernel_size is one of the hyperparameters that you will be setting when building a convolutional layer. Like most neural network hyperparameters, no single best answer fits all problems. The intuition is that smaller filters will capture very fine details of the image, and bigger filters will miss minute details in the image.
- **STRIDES AND PADDING:**
  - **Strides**—The amount by which the filter slides over the image. For example, to slide the convolution filter one pixel at a time, the strides value is 1. If we want to jump two pixels at a time, the strides value is 2. Strides of 3 or more are uncommon and rare in practice. Jumping pixels produces smaller output volumes spatially. Strides of 1 will make the output image roughly the same height and width as the input image, while strides of 2 will make the output image roughly half of the input image size. I say "roughly" because it depends on what you set the padding parameter to do with the edge of the image.

- **Padding**—Often called zero-padding because we add zeros around the border of an image (figure 5.13). Padding is most commonly used to allow us to preserve the spatial size of the input volume so the input and output width and height are the same. This way, we can use convolutional layers without necessarily shrinking the height and width of the volumes. This is important for building deeper networks, since, otherwise, the height/width would shrink as we went to deeper layers.

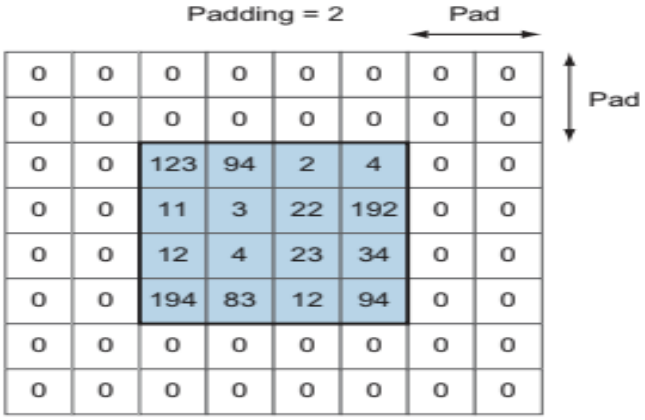


Figure 5-13 Zero-padding adds zeros around the border of the image. Padding = 2 adds two layers of zeros around the border.

When using strides and padding hyperparameters, the objective is to either preserve all of the crucial image details and pass them on to the following layer (when the strides value is 1 and the padding value is the same) or to ignore some of the spatial details of the image to reduce the processing's computational cost. To focus on the retrieved features, we will be reducing the size of the image by adding the pooling layer (described later). For the time being, be aware that strides and padding hyperparameters are used to regulate the convolutional layer's output size and behavior, determining whether to pass on all picture details or to disregard some of them.

1.7.4. Pooling layers or subsampling

The depth of the output layer rises as more convolutional layers are added, increasing the number of parameters the network must optimize (learn). You can see that increasing the number of convolutional layers will result in a significant number of parameters (often tens or even hundreds) (weights). The time and spatial complexity of the mathematical operations that take place throughout the learning process rise due to the increase in network dimensionality. Pooling layers are very handy in this situation. By lowering the number of parameters given to the following layer, subsampling or pooling aids in network size reduction. Input is resized by the pooling operation using a summary statistical function, such as a maximum or average, to reduce the overall number of parameters passed on to the next layer. The pooling layer's objective is to reduce the computational complexity by down sampling the convolutional layer's feature maps into a

smaller set of parameters. In the CNN design, it is customary to add pooling layers every one to two convolutional layers.

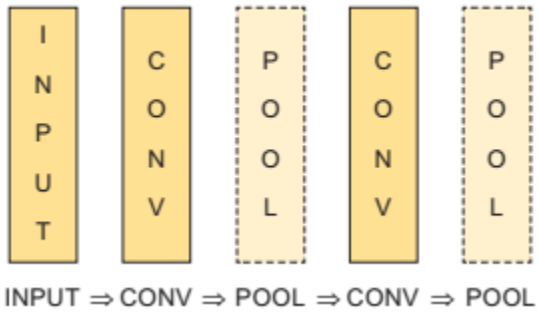


Figure 5-14 Pooling layers are commonly added after every one or two convolutional layers.

- **MAX POOLING VS. AVERAGE POOLING**
  - **Max-Pooling:** The difference with max-pooling is that the windows don't have weights or any values. All they do is slide over the feature map created by the previous convolutional layer and select the max pixel value to pass along to the next layer, ignoring the remaining values. In figure 5.15, you see a pooling filter with a size of 2 × 2 and strides of 2 (the filter jumps 2 pixels when sliding over the image). This pooling layer reduces the feature map size from 4 × 4 down to 2 × 2.

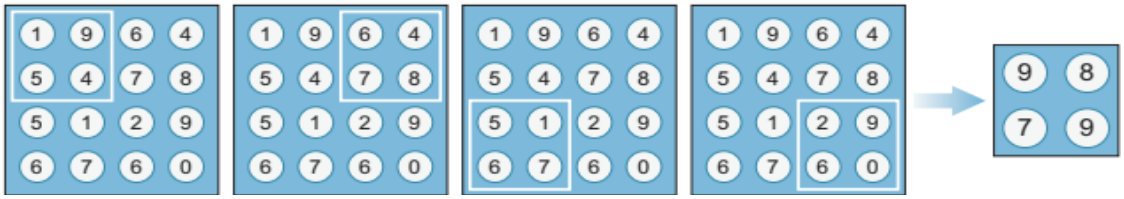


Figure 5-15 A 2 × 2 pooling filter and strides of 2, reducing the feature map from 4 × 4 to 2 × 2

When we do that to all the feature maps in the convolutional layer, we get maps of smaller dimensions (width times height), but the depth of the layer is kept the same because we apply the pooling filter to each of the feature maps from the previous filter. So if the convolutional layer has three feature maps, the output of the pooling layer will also have three feature maps, but of a smaller size (figure 5.16).

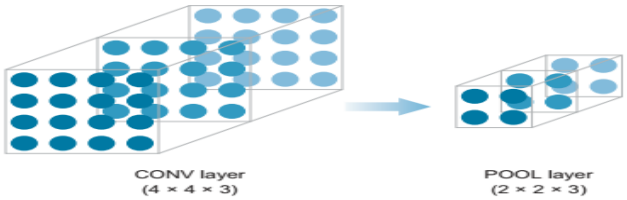


Figure 5-16 If the convolutional layer has three feature maps, the pooling layer's output will have three smaller feature maps.

- **Global average pooling:** layer takes a 3D array and turns it into a vector.

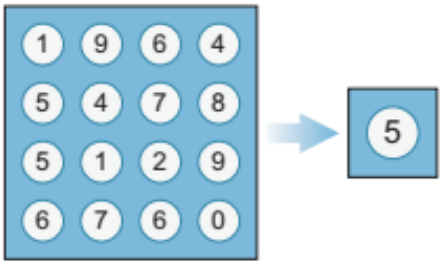


Figure 5-17 Global average pooling calculates the average values of all the pixels in a feature map

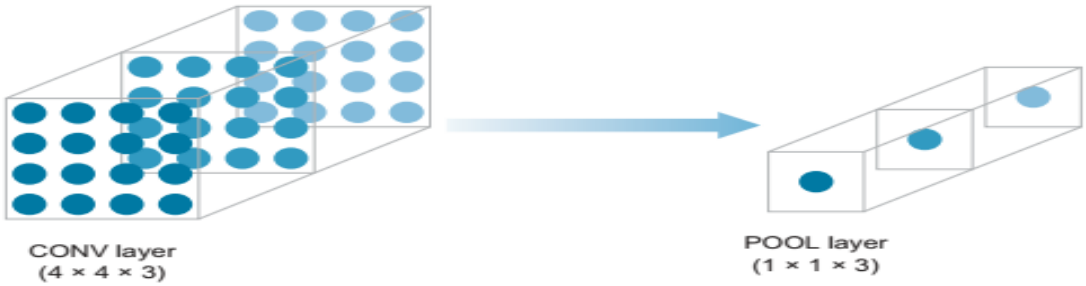


Figure 5-18 The global average pooling layer turns a 3D array into a vector.

WHY USE A POOLING LAYER?

- As you can see from the examples we've covered, pooling layers causes our convolutional layers to become less dimensional. Reduced dimensionality is crucial since CNNs may have multiple convolutional layers and tens or hundreds of convolutional filters per layer in large applications (kernels). Since the parameters (weights) that the network learns are contained in the kernel, this can easily spiral out of control and cause the dimensionality of our convolutional layers to increase significantly. Therefore, increasing pooling layers reduces the dimensionality of the image while preserving the relevant features and passing them on to the next layer. Pooling layers can be compared to image-compression software. They scale back the image resolution while preserving its key elements.

➢ Recap

Up until this point, we used a series of convolutional and pooling layers to process an image and extract meaningful features that are specific to the images in the training dataset. To summarize how we got here:

1. The raw image is fed to the convolutional layer, which is a set of kernel filters that slide over the image to extract features.
2. The convolutional layer has the following attributes that we need to configure:

```
from keras.layers import Conv2D
model.add(Conv2D(filters=16, kernel_size=2,
strides='1',
padding='same', activation='relu'))

from keras.layers import MaxPooling2D
model.add(MaxPooling2D(pool_size=(2, 2),
strides = 2))
```

Filters: is the number of kernel filters in each layer (the depth of the hidden layer).

kernel_size: is the size of the filter (aka kernel). Usually 2, or 3, or 5.

Strides are the amount by which the filter slides over the image. A strides value of 1 or 2 is usually recommended as a good start.

Padding: adds columns and rows of zero values around the border of the image to reserve the image size in the next layer.

activation of **relu** is strongly recommended in the hidden layers.

6. Mask Detection using CNN

1. Mask Detection

1.1. Objective

Effective COVID-19 pandemic control methods require close attention to reduce the harmful effects on global economic and public health, with a full future still ahead. Numerous methods are advised by WHO to reduce infection rates and prevent depleting the available medical resources in the absence of efficient antivirals. One non-pharmaceutical intervention strategy that can be utilized to reduce the main source of SARS-CoV2 droplets released by an infected person is wearing a mask. All nations now require masks over the mouth and nose when in public, regardless of debates over medical resources and the variety of masks available. After detecting whether a person is wearing a mask or not we can control to open the gate and let the person pass. The model set a flag in a real-time database on the firebase cloud and using **Node-MCU** we read the flag and take the action.

1.2. System Architecture and block diagram

We built the Deep Learning model Using the CNN Architecture and we have used the transfer learning using **MobileNetV2.** we got high accuracy of about 99% with the dataset we have.

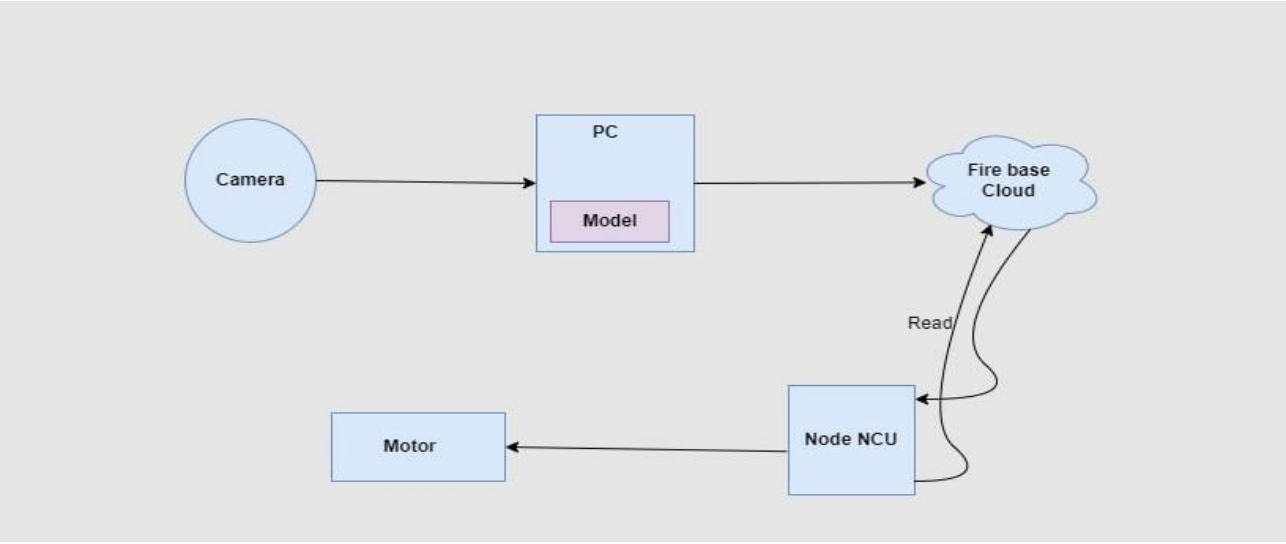


Figure 6-6-1 System Architecture

1.3.**Model Architecture**

In fact, we have built several models during the experimentation process trying to get a model with better results. We got a model with an accuracy of 99% and low losses.

1.3.1. **Baseline Model**

- **Model Architecture:**
  - ✓ The model consists of Convolution Layers with activation function **relu**.
  - ✓ After each CONV layer, we have a Max-Pooling layer.
  - ✓ Then we add a Dropout layer to limit overfitting.
  - ✓ Then Flatten all layers and convert them to a vector.
  - ✓ We use Dense layers to classify the Images.

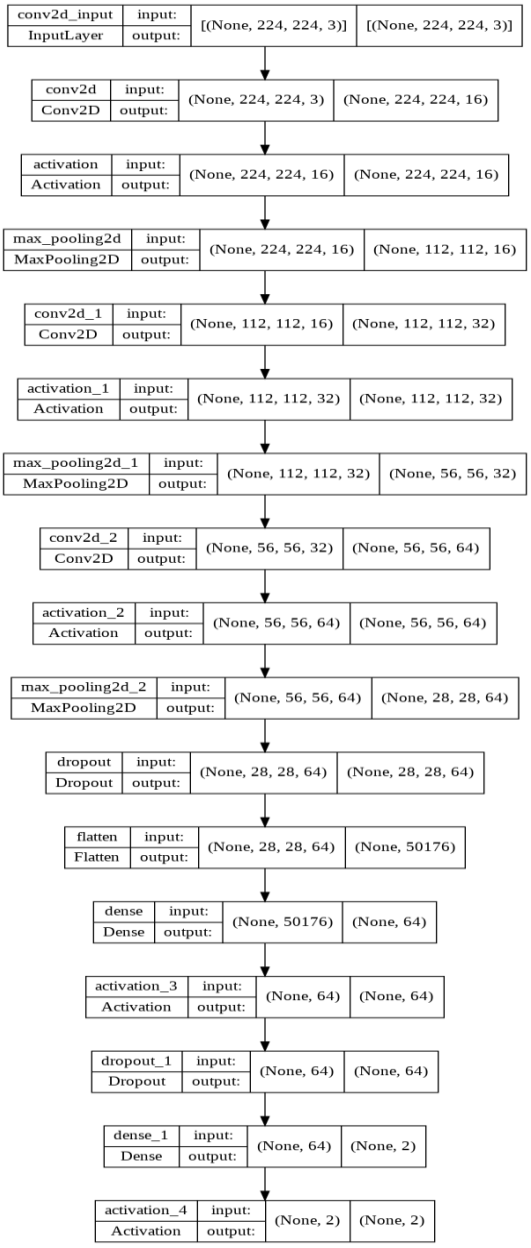


Figure 6-2 Baseline Model Architecture.

- **Model Training and losses figure**

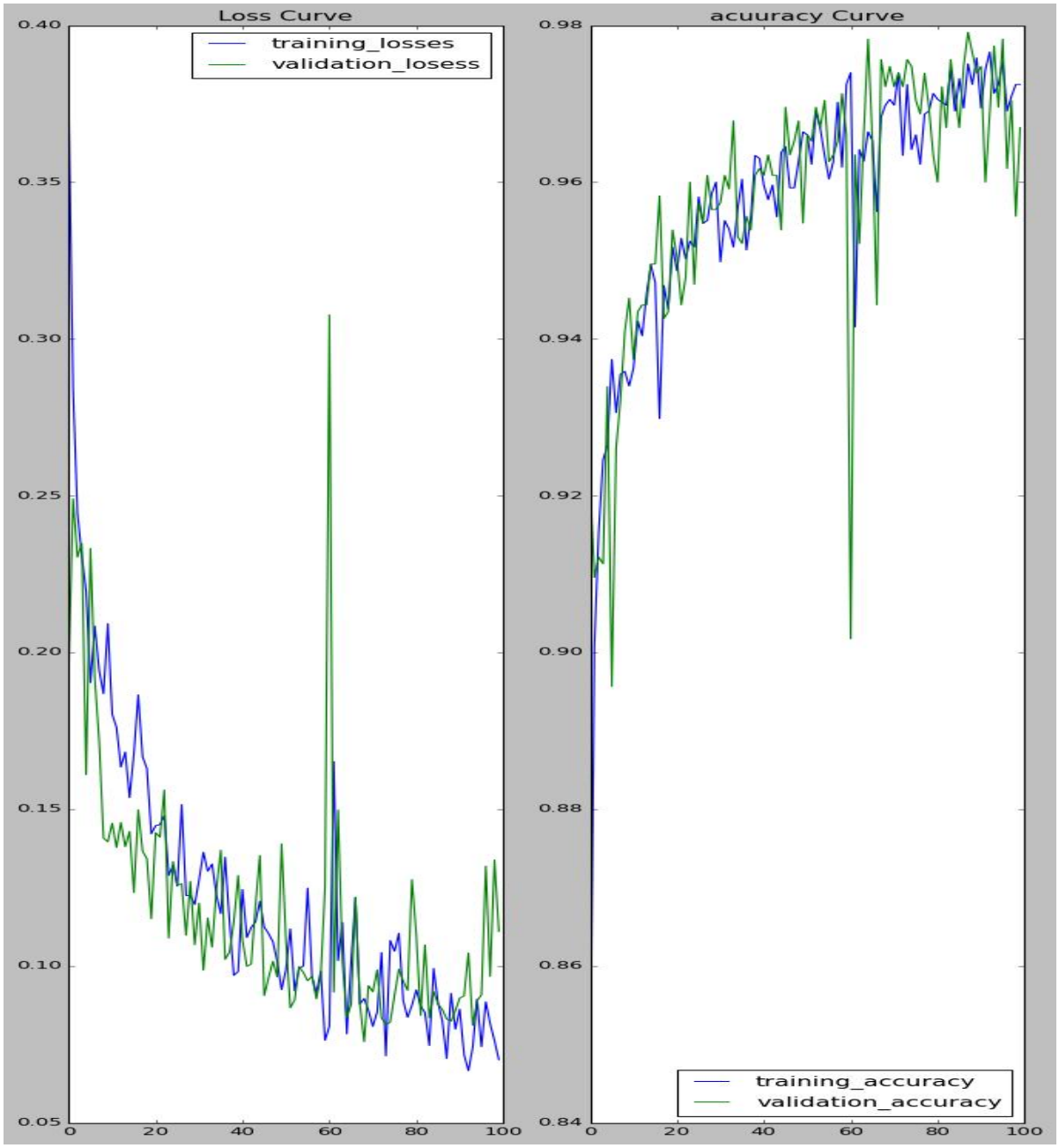


Figure 6-6-3 Baseline Model Train loss figure.

- **Model ROC Curve**

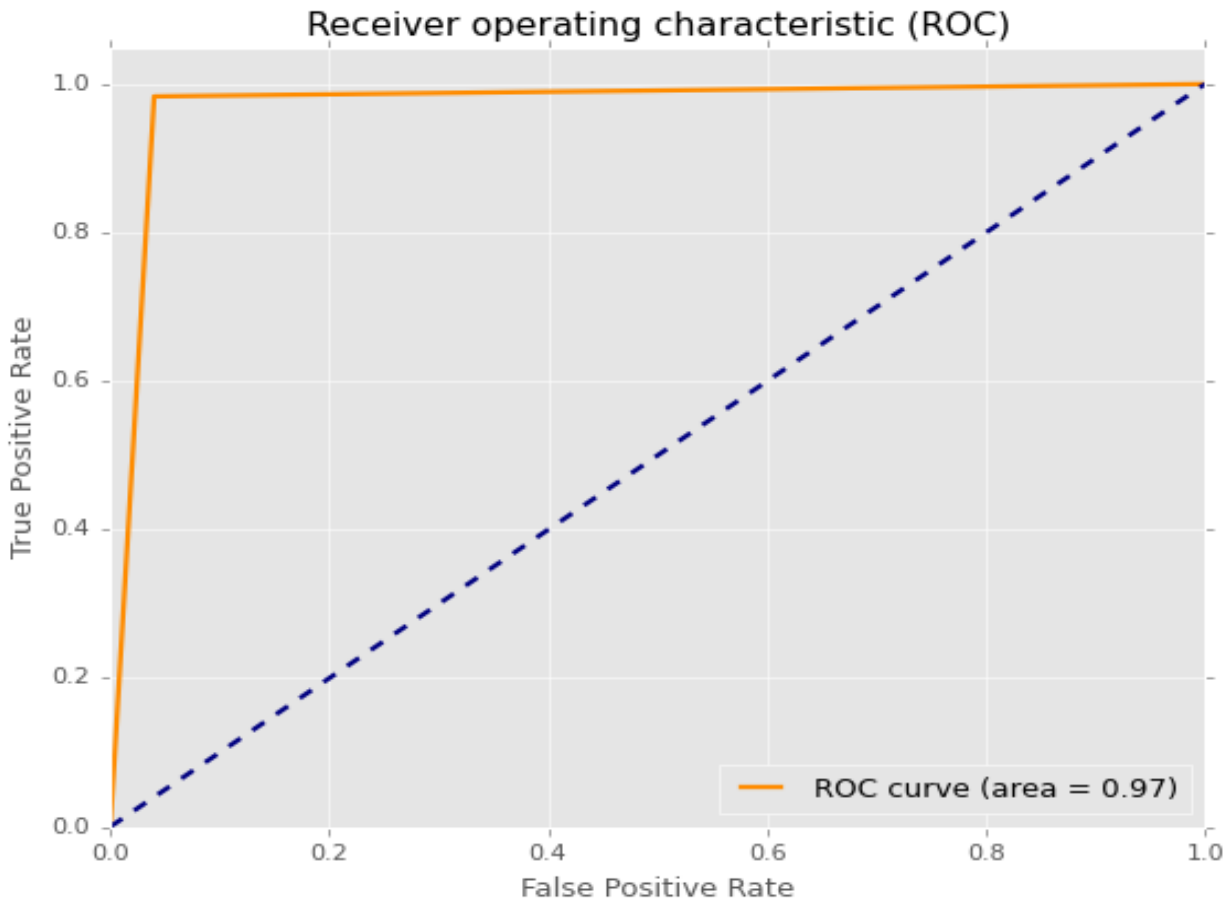


Figure 6-4 Model Baseline ROC Curve.

- **Confusion Matrix**

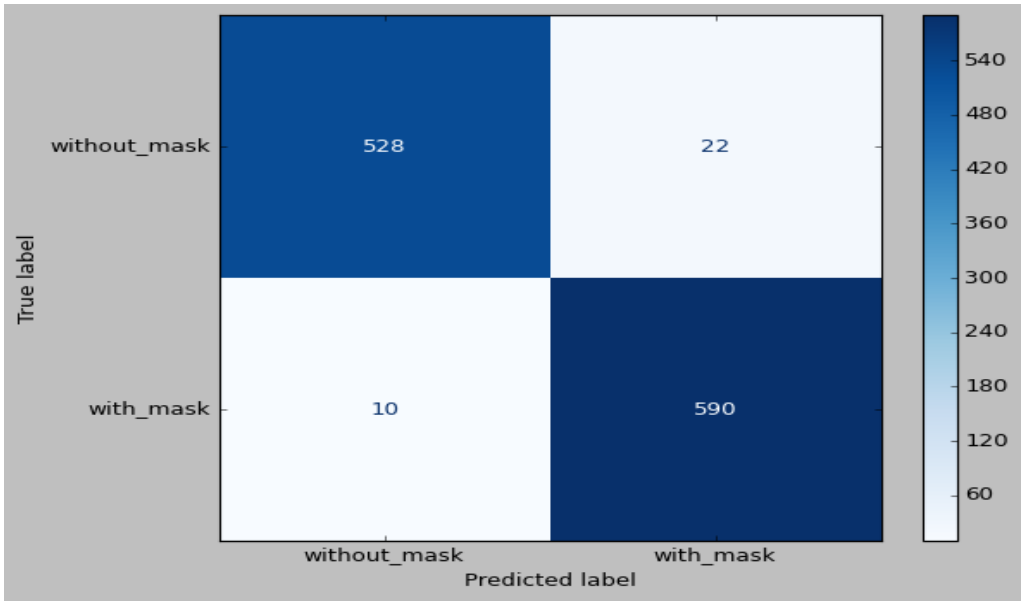


Figure 6-5 Baseline Confusion Matrix

1.3.2. Enhanced Model

- **Model Architecture:**
  - ✓ The model consists of Convolution Layers with activation function **relu** but increased the number of filters and used **SGD** optimizer instead of **Adam** optimizer.
  - ✓ After each **CONV** layer, we have a **Max-Pooling** layer.
  - ✓ Then we add a **Dropout** layer to limit overfitting.
  - ✓ Then **Flatten** all layers and convert them to a vector.
  - ✓ We use **Dense** layers to classify the Images.

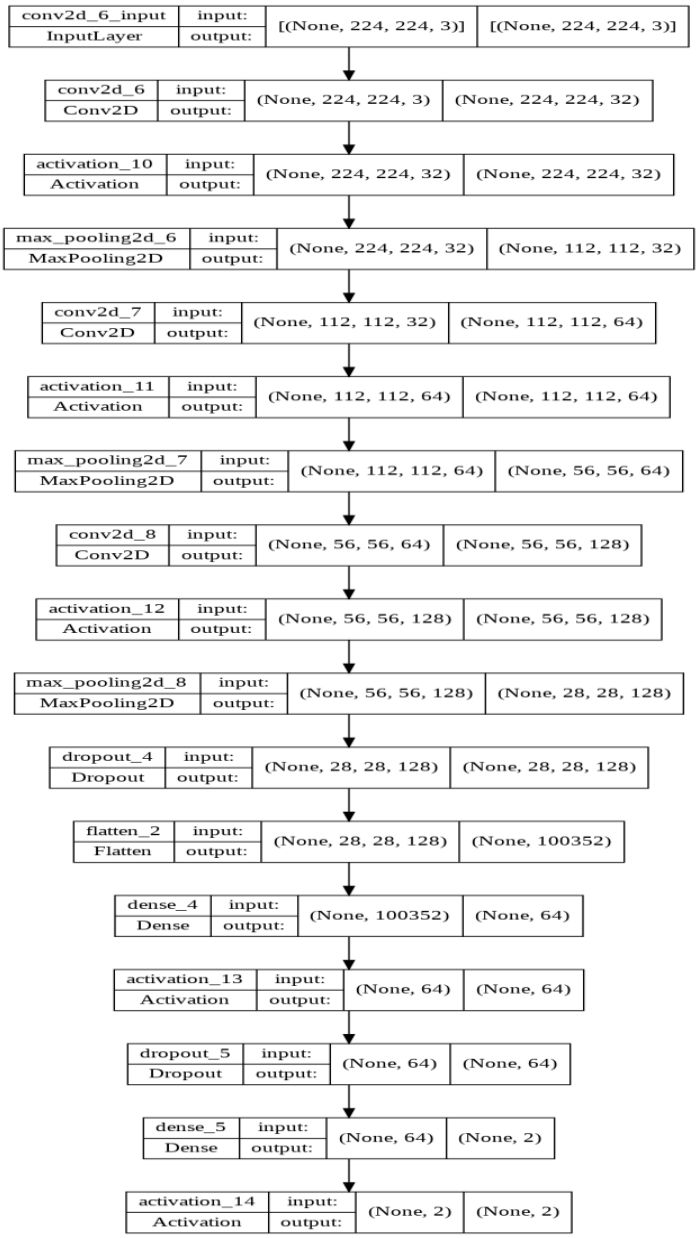


Figure 6-6 Enhanced Model Architecture

- **Model Training and losses figure**

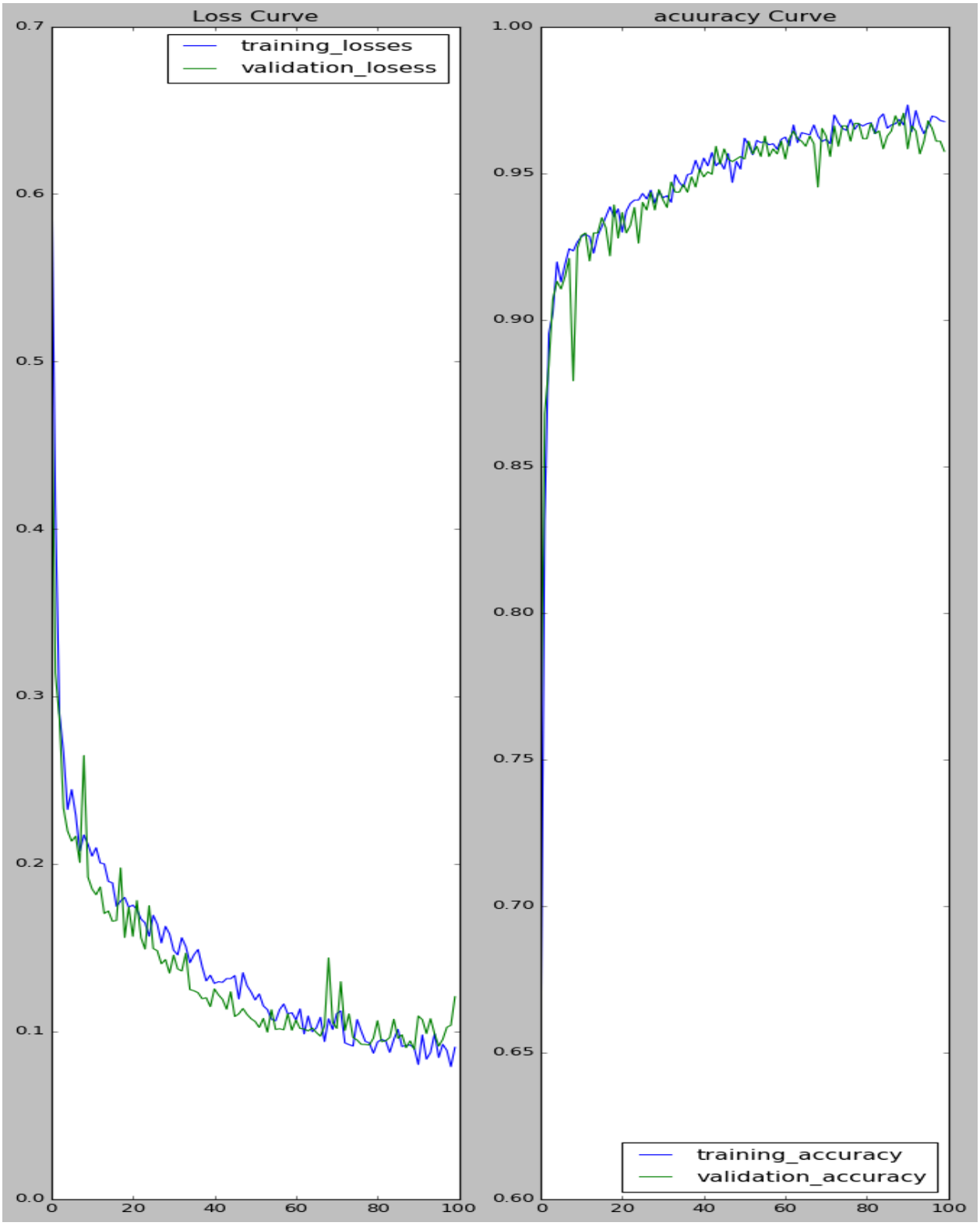


Figure 6-7 Enhanced Model Train and Loss Figure

- **Enhanced Model ROC Curve**

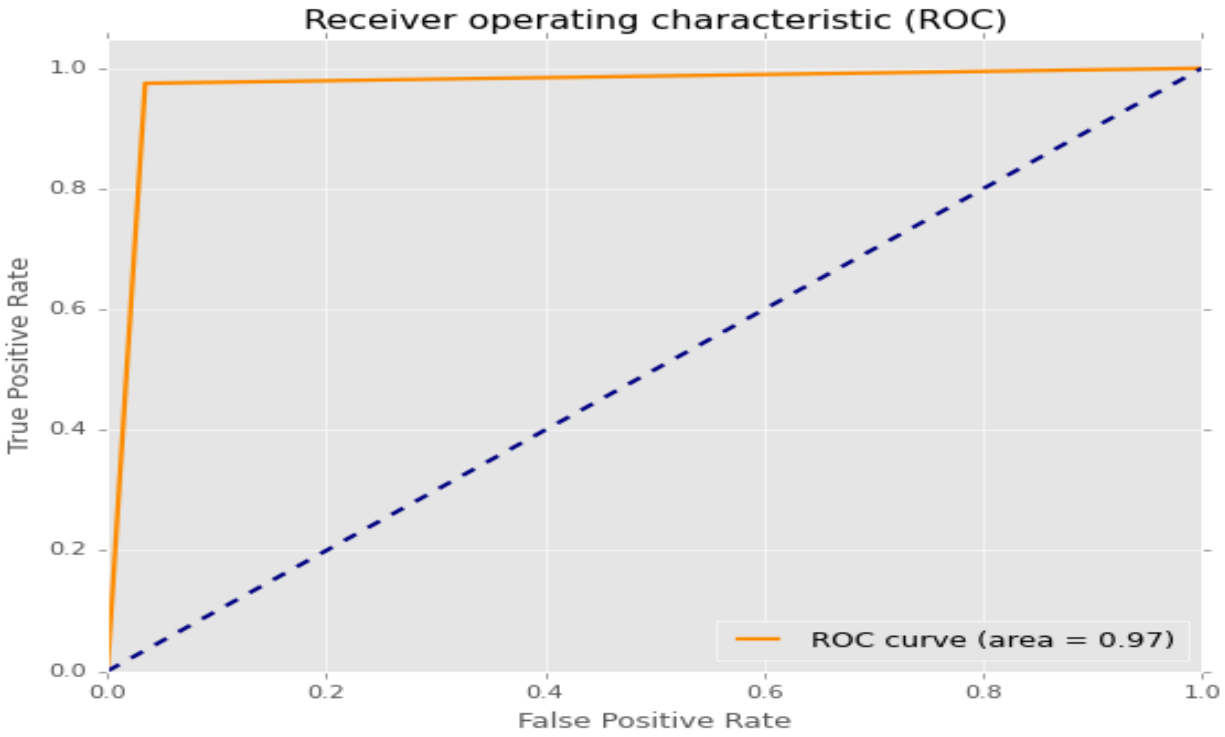


Figure 6-8 Enhanced model ROC curve

- **Enhanced Model Confusion Matrix**

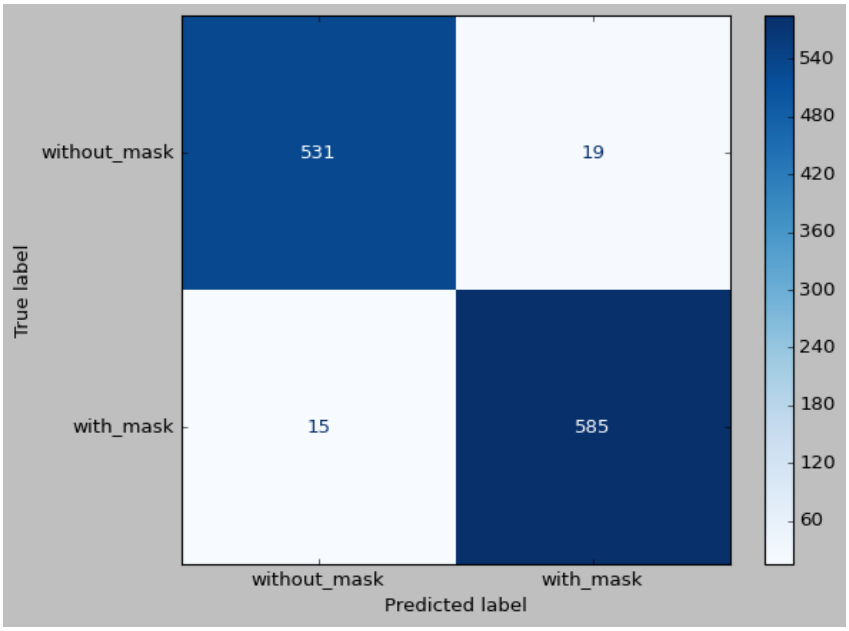


Figure 6-9 Enhanced Model Confusion Matrix

1.3.3. **Final Model**

- **Model Architecture**
  - **3 CONV** Layers with **Relu** activation function and the kernel size=3.
  - After each **CONV** layer, there is a **max-Pooling** Layer. Then **Dropout** Layer.
  - **3 CONV** layers with **Relu** activation function and the kernel size=3.
  - After each **CONV** layer, there is a **max-Pooling** Layer. Then **Dropout** Layer.
  - Then **Flatten** all layers and convert them to a vector. We use **Dense** layers to classify the Images.

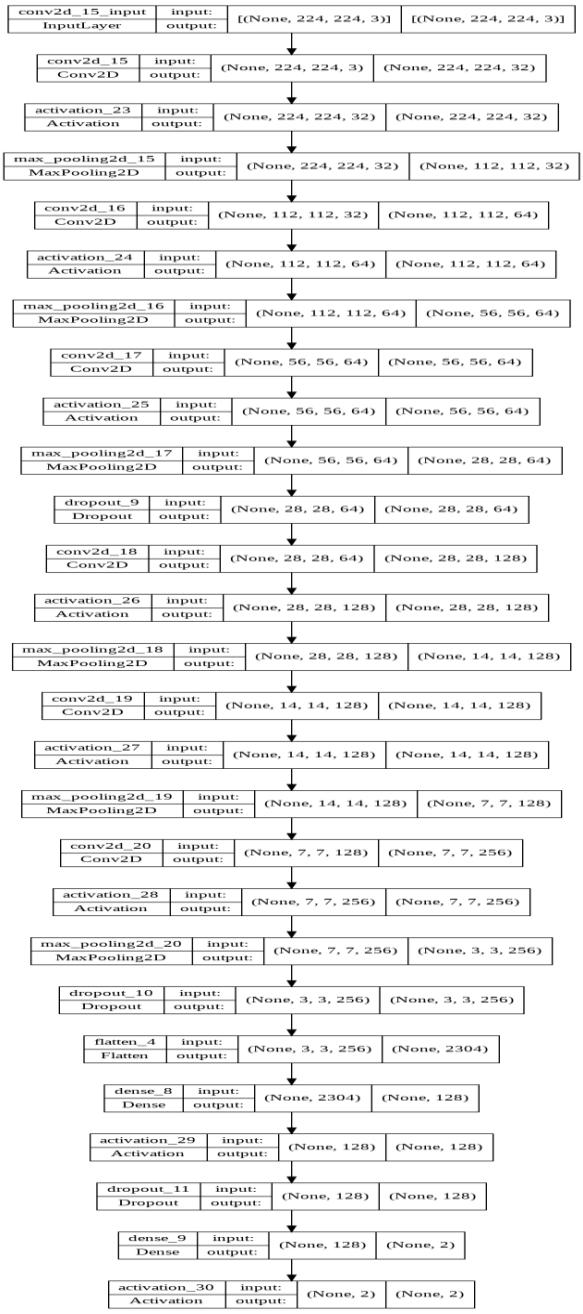


Figure 6-10 Final Model Architecture

- **Model Training and Loss figure**

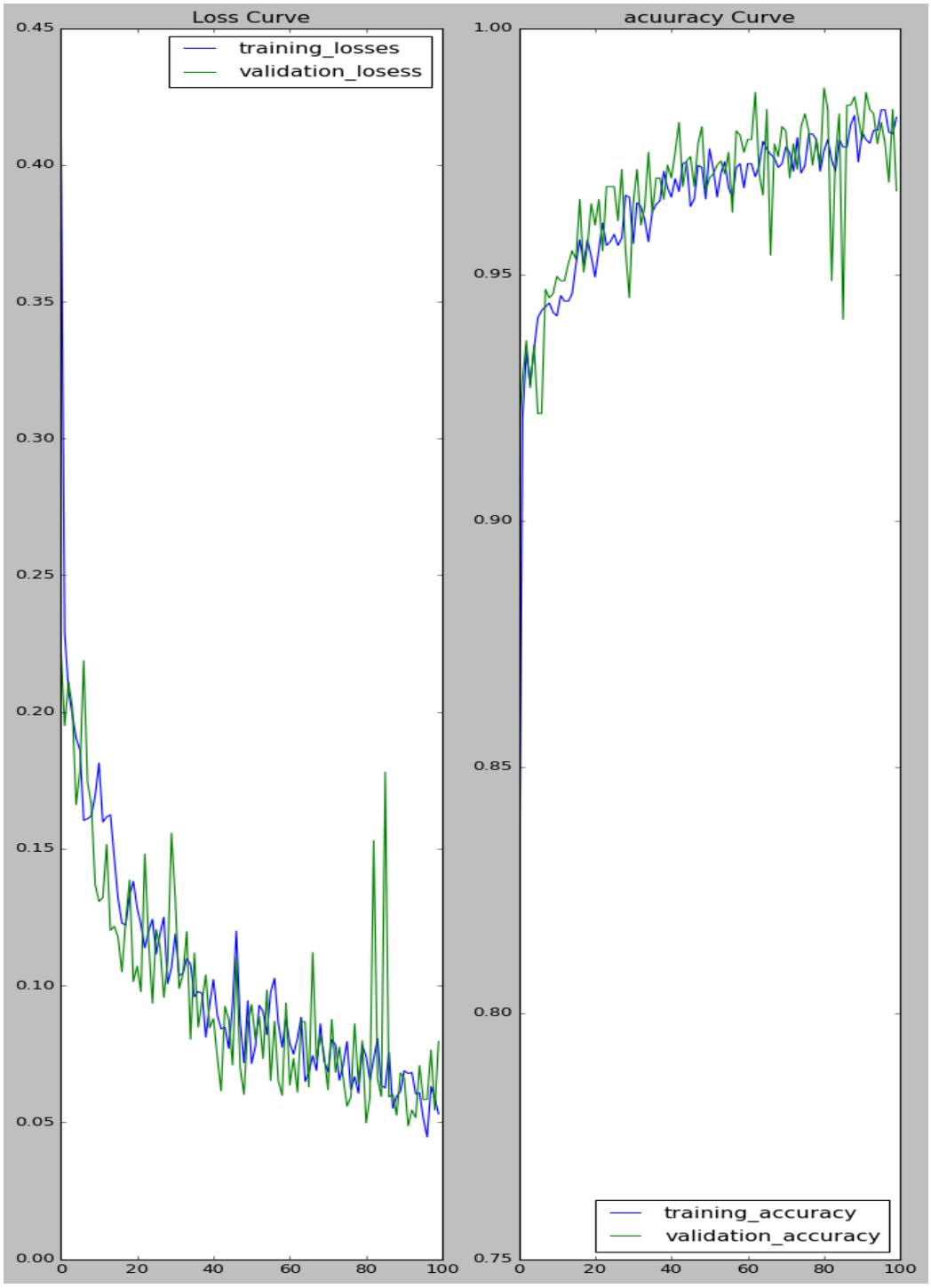


Figure 6-11 Final Model Train and Loss curve

- **Final Model ROC Curve**

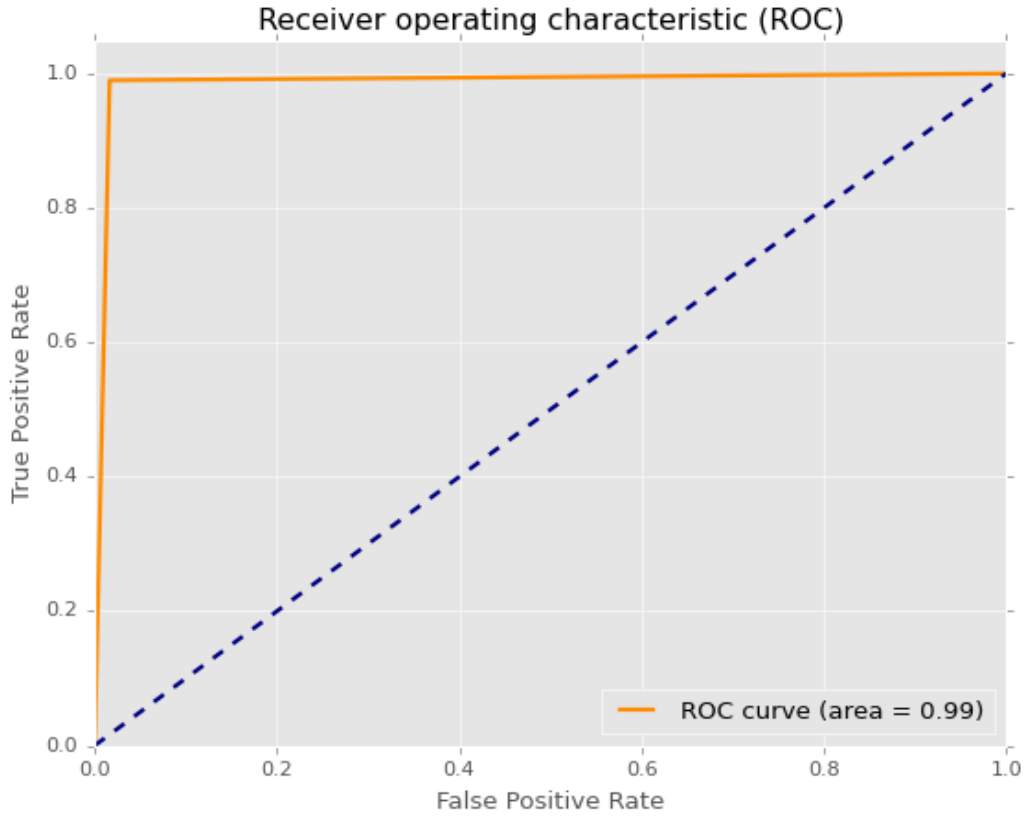


Figure 6-12 Final Model ROC Curve

- **Final Model Confusion Matrix**

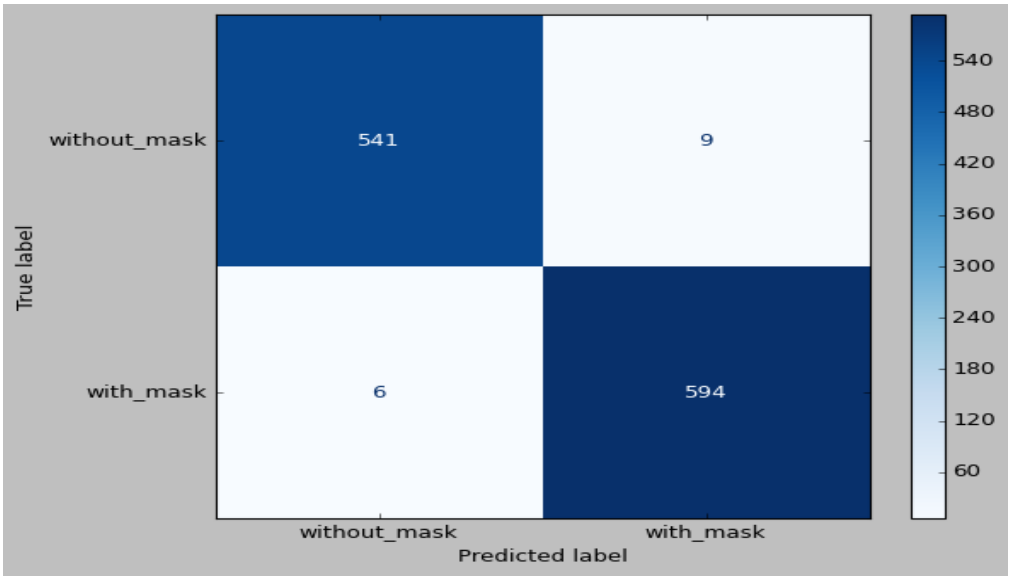


Figure 6-13 Final Model Confusion Matrix

1.4. System Flow Chart

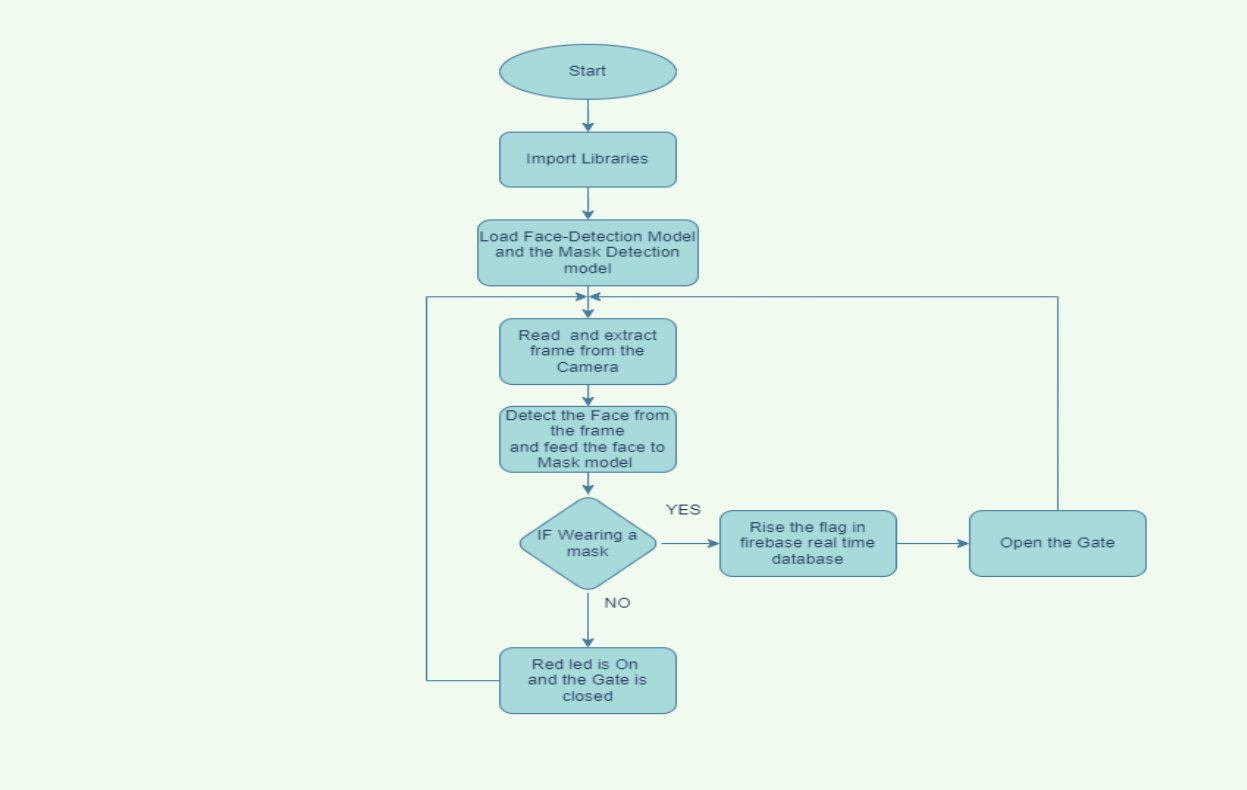


Figure 6-14 System Flow Chart.

1.5. Pseudo Code

- import the necessary packages.
- load our serialized face detector model from the disk.
- load the face mask detector model from the disk.
- **Loop Forever:**
    - grab the frame from the threaded video stream and resize it.
    - detect faces in the frame and determine if they are wearing a face mask or not.
    - loop over the detected face locations and their corresponding.
    - determine the class label and color we'll use to draw.
    - display the label and bounding box rectangle on the output.
    - show the output frame.
    - IF Mask is detected;
        - Update the Access Flag in the real-time database of Firebase to True.
    - Else:
        - Update the Access Flag to False.
    - Using Node MCU read the Access Flag
    - IF True:

        - Open the Door
        - Green Led ON
    - Else
        - Close the Door
        - Red Led ON

1.6. Firebase Real-Time database

Firebase is a powerful platform for your mobile and web application. Firebase can power your app's backend, including data storage, user authentication, static hosting, and more. Firebase lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your servers. It's supporting for Android, iOS, C++, Unity, and Web Platform. Firebase is a Google platform, free and easy to use, it has a real-time database that we will explain later what we need it for.

1.7. Using a Real-time database in our project

We discuss in the last part that we need the file URL to download it, so we can't include the URL in our code every time we need to update since the Node MCU will be installed in our vehicle. That's why we need to use a real-time database to send a notification to the Node MUU that there is a new update and also send the URL needed to download the file let's see how we can do this in the next steps. Let's explain this procedure in a simple example, we have a file uploaded on firebase cloud storage, so we have the file's URL. First, we will create a real-time database from the left bar after creating a firebase account as shown in Figures (6-15).

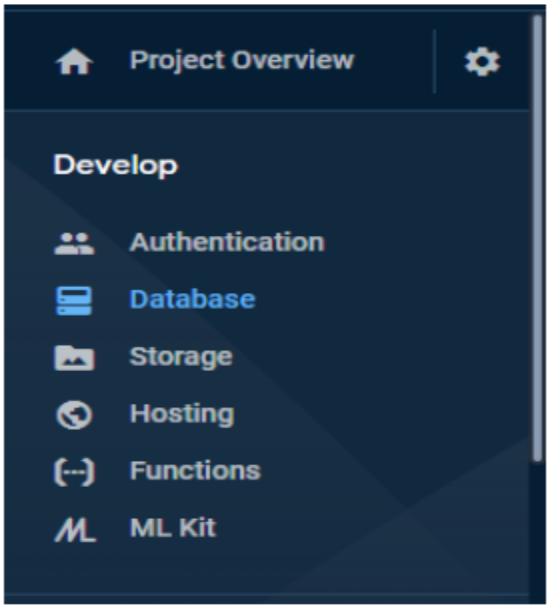


Figure 6-15 Firebase create a real time database.

Second, we will create records in this database as we want, and shown in the next picture for our example, we have a flag that tells Node MCU to start receiving the URL to start downloading the default will be **—false** and then we will change it to **—true** as shown in Figure (6-16).

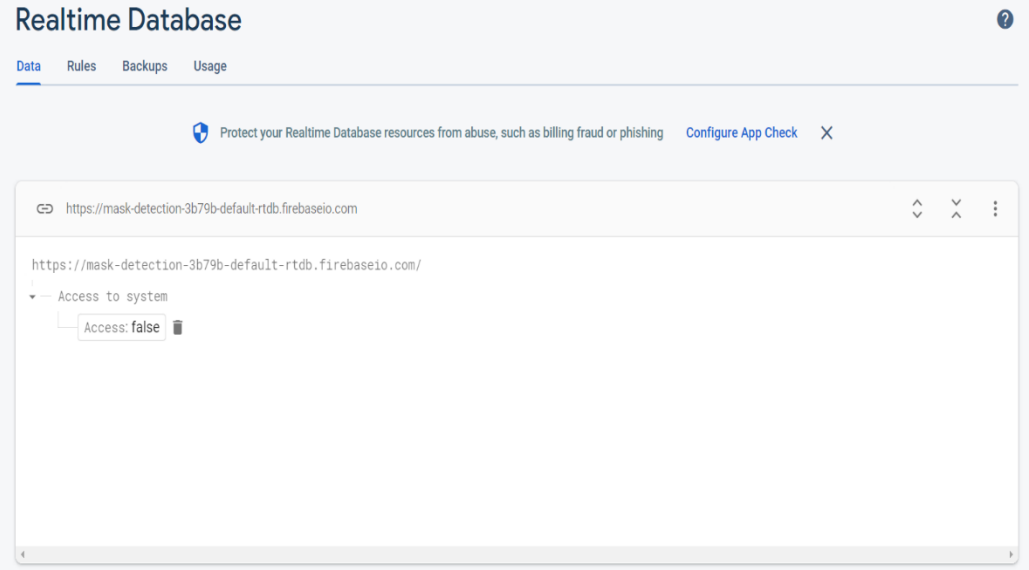


Figure 6-16 Database record and its initial value

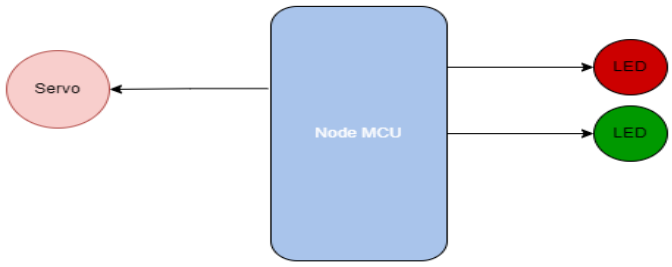


Figure 6-17 Node MCU Connection

7. Chapter 7: Auto Attendance System using CNN Face Recognition

1. Objective

Face recognition is a very useful algorithm that can be used for a large number of applications such as person verification, Door Lock system, and in our project, we are going to use the Face recognition algorithm for recording the attendance of students in a classroom. The idea of the project is very helpful so the teacher or a doctor will have an excel sheet with the student attending the lecture. Face recognition is a very common computer vision application that needs previous knowledge of Open-CV for image processing, Keras Api, numPy, and matplotlib for visualization.

2. Tools and Software

2.1. **Keras**

- Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.
- Keras is the most used deep learning framework among the top-5 winning teams on Kaggle. Because Keras makes it easier to run new experiments, it empowers you to try more ideas than your competition, faster. And this is how you win.
- Built on top of TensorFlow 2, Keras is an industry-strength framework that can scale to large clusters of GPUs or an entire TPU pod. It's not only possible; it's easy.
- Take advantage of the full deployment capabilities of the TensorFlow platform. You can export Keras models to JavaScript to run directly in the browser, to TF Lite to run on iOS, Android, and embedded devices. It's also easy to serve Keras models via a web API.

```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video.encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video question answering model:
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model(inputs=[video_input, question_input],
                             outputs=output)
```

Figure 7-1 Keras Example

2.2. **OpenCV**

- OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

2.3. **FaceNet**

- **FaceNet** is a face recognition system developed in 2015 by researchers at Google that achieved then state-of-the-art results on a range of face recognition benchmark datasets. The **FaceNet** system can be used broadly thanks to multiple third-party open-source implementations of the model and the availability of pre-trained models.
- The **FaceNet** system can be used to extract high-quality features from faces, called face embeddings, that can then be used to train a face identification system.

2.3.1. **How does FaceNet work?**

- FaceNet takes an image of the person's face as input and outputs a vector of 128 numbers which represent the most important features of a face. In machine learning, this vector is called embedding. Why embedding? Because all the important information from an image is embedded into this vector. **FaceNet** takes a person's face and compresses it into a vector of 128 numbers. Ideally, embeddings of similar faces are also similar.

- Mapping high-dimensional data (like images) into low-dimensional representations (embeddings) has become a fairly common practice in machine learning these days.
- Ok, what do we do with these embeddings? How do we recognize a person using an embedding?
- Embeddings are vectors and we can interpret vectors as points in the Cartesian coordinate system. That means we can plot an image of a face in the coordinate system using its embeddings.
- **FaceNet** learns in the following way:
    - Randomly selects an anchor image.
    - Randomly select an image of the same person as the anchor image (positive example).
    - Randomly select an image of a person different than the anchor image (negative example).
    - Adjusts the **FaceNet** network parameters so that the positive example is closer to the anchor than the negative example.
    - We repeat these steps until there are no more changes to be done all the faces of the same person are close to each other and far from the others.
    - This method of learning with anchor, positive and negative examples is called triplet loss.

3. Face Recognition workflow

- **Face detection** — Detecting one or more faces in an image.
- **Feature extraction** — Extracting the most important features from an image of the face.
- **Face classification** — Classifying the face based on extracted features.

  There are various ways to implement each of the steps in a face recognition pipeline. In this chapter, we'll focus on popular deep learning approaches where we perform face detection using **MTCNN**, feature extraction using **FaceNet,** and classification using **Softmax**.

Figure 7-2 Face Recognition Pipeline.

- **MTCN** or Multi-Task Cascaded Convolutional Neural Networks is a neural network that detects faces and facial landmarks on images. It was published in 2016 by Zhang et al.

4. System Design and Analysis

4.1. Block Diagram

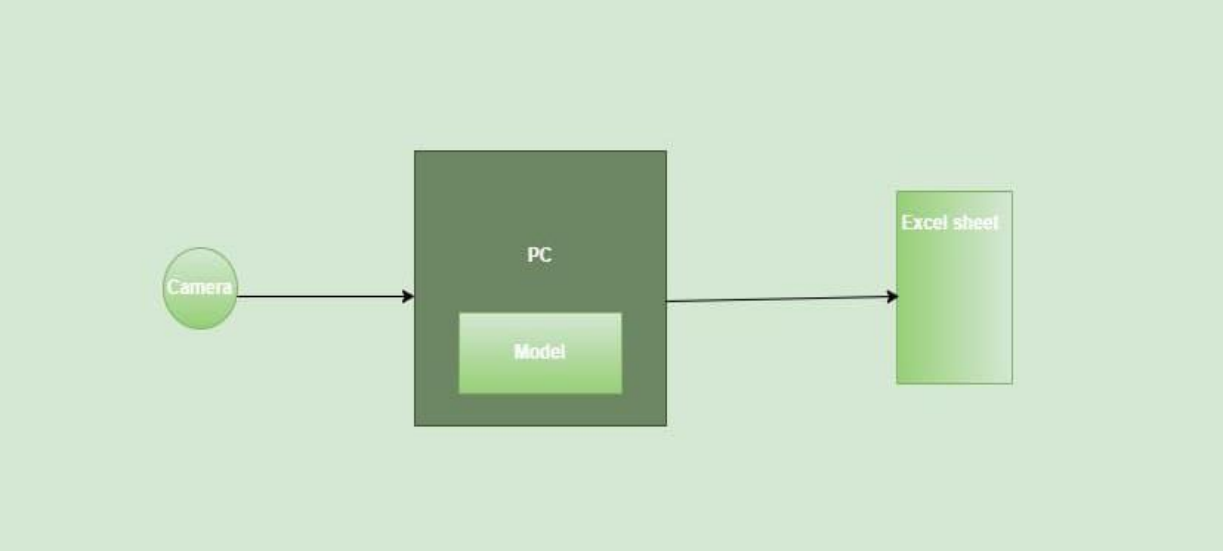


Figure 7-3 System Block Diagram.

4.2. Flow Chart

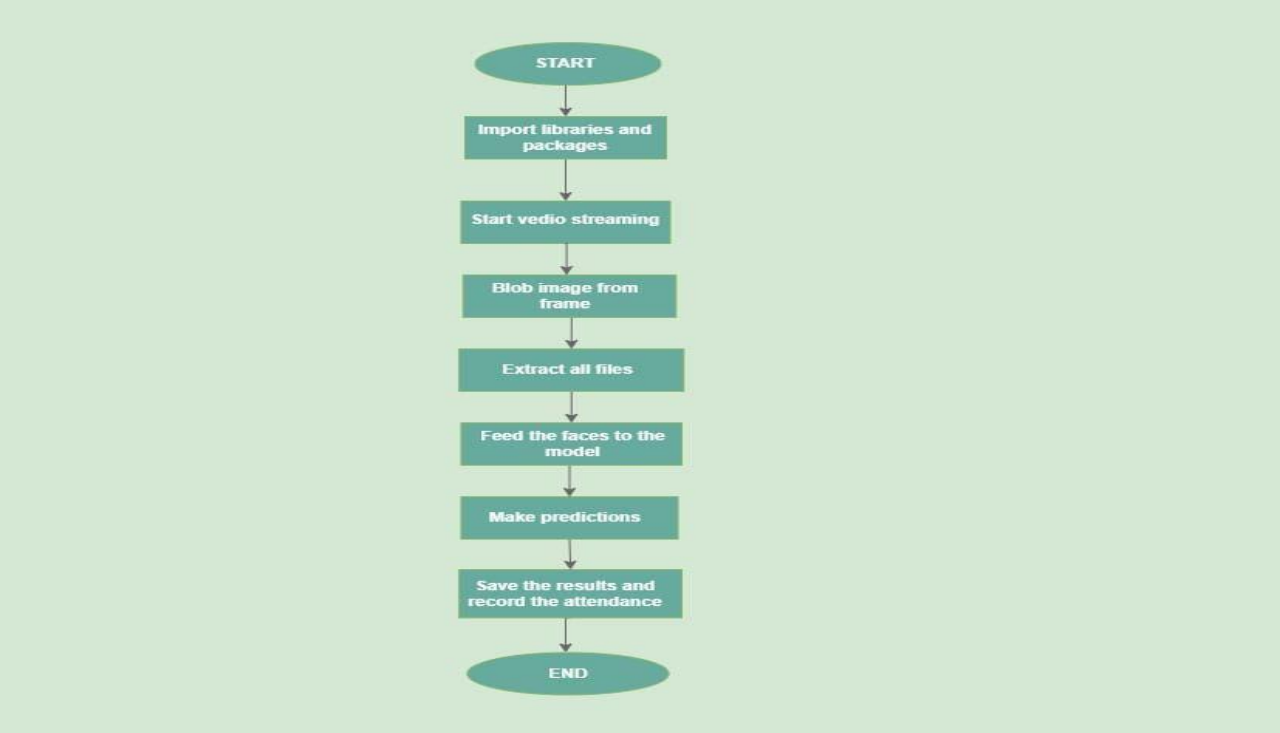


Figure 7-4 System Application Flow Chart.

5. Face Recognition

5.1. Detect Faces

- The first step is to detect the face in each photograph and reduce the dataset to a series of faces only.

```
from os import listdir
```

```
from PIL import Image
from numpy import asarray
from matplotlib import pyplot
from mtcnn.mtcnn import MTCNN

# extract a single face from a given photograph
def extract_face(filename, required_size=(160, 160)):
  # load image from file
  image = Image.open(filename)
  # convert to RGB, if needed
  image = image.convert('RGB')
  # convert to array
  pixels = asarray(image)
  # create the detector, using default weights
  detector = MTCNN()
  # detect faces in the image
  results = detector.detect_faces(pixels)
  if results:
    # extract the bounding box from the first face
    x1, y1, width, height = results[0]['box']
    # bug fix
    x1, y1 = abs(x1), abs(y1)
    x2, y2 = x1 + width, y1 + height
    # extract the face
    face = pixels[y1:y2, x1:x2]
    # resize pixels to the model size
    image = Image.fromarray(face)
    image = image.resize(required_size)
    face_array = asarray(image)
    return face_array
```

5.2. Create Face Embeddings

- The next step is to create a face embedding. A face embedding is a vector that represents the features extracted from the face. This can then be compared with the vectors generated for other faces. For example, another close vector (by some

measure) may be the same person, whereas another vector that is far (by some measure) may be a different person.

- The classifier model that we want to develop will take a face embedding as input and predict the identity of the face. The **FaceNet** model will generate this embedding for a given image of a face.
- The **FaceNet** model can be used as part of the classifier itself, or we can use the **FaceNet** model to pre-process a face to create a face embedding that can be stored and used as input to our classifier model. This latter approach is preferred as the **FaceNet** model is both large and slow to create a face embedding.
- We can, therefore, pre-compute the face embeddings for all faces in the train and test (formally 'Val') sets in our Faces Dataset.

```
# calculate a face embedding for each face in the dataset using facenet
```

```
from numpy import load
from numpy import expand_dims
from numpy import asarray
from numpy import savez_compressed
from keras.models import load_model
import numpy as np
# get the face embedding for one face
def get_embedding(model, face_pixels):
  # scale pixel values
  face_pixels = np.float32(face_pixels)
  # standardize pixel values across channels (global)
  mean, std = face_pixels.mean(), face_pixels.std()
  face_pixels = (face_pixels - mean) / std
  # transform face into one sample
  samples = expand_dims(face_pixels, axis=0)
  # make prediction to get embedding
  yhat = model.predict(samples)
  return yhat[0]
```

5.3. Perform Face Classification

- First, it is a good practice to normalize the face embedding vectors. It is a good practice because the vectors are often compared to each other using a distance metric.
- In this context, vector normalization means scaling the values until the length or magnitude of the vectors is 1 or unit length. This can be achieved using the Normalizer class in scikit-learn. It might even be more convenient to perform this step when the face embeddings are created in the previous step.
- Next, the string target variables for each celebrity name need to be converted to integers.
- This can be achieved via the **LabelEncoder** class in **scikit-learn.**

- It is common to use a Linear Support Vector Machine (SVM) when working with normalized face embedding inputs. This is because the method is very effective at separating the face embedding vectors. We can fit a linear SVM to the training data using the SVC class in scikit-learn and setting the '*kernel*' attribute to '*linear*'. We may also want probabilities later when making predictions, which can be configured by setting '*probability*' to '*True*'.

```
from numpy import load
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Normalizer
from sklearn.svm import SVC
import pickle
# load dataset
data = load('/content/drive/MyDrive/Face_recognition/Dataset-faces-
embeddings.npz', allow_pickle=True)
trainX, trainy, valX, valy, testX, testy = data['arr_0'], data['arr_1'], data['ar
r_2'], data['arr_3'] ,  data['arr_4'] , data['arr_5']
print('Dataset: train=%d, val=%d, test=%d' % (trainX.shape[0],valX.shape[0],testX
.shape[0]))
# normalize input vectors
in_encoder = Normalizer(norm='l2')
trainX = in_encoder.transform(trainX)
testX = in_encoder.transform(testX)
valX = in_encoder.transform(valX)
# label encode targets
out_encoder = LabelEncoder()
out_encoder.fit(trainy)
trainy = out_encoder.transform(trainy)
testy = out_encoder.transform(testy)
valy = out_encoder.transform(valy)
# fit model
model = SVC(kernel='linear', probability=True , verbose=1, max_iter=1000)
model.fit(trainX, trainy)
# predict
yhat_train = model.predict(trainX)
yhat_test = model.predict(testX)
yhat_val = model.predict(valX)
# score
score_train = accuracy_score(trainy, yhat_train)
score_test = accuracy_score(testy, yhat_test)
score_val = accuracy_score(testy, yhat_test)
# summarize
print('Accuracy: train=%.3f,val=%.3f, test=%.3f' % (score_train*100,score_val*100
, score_test*100))
```

8. Chapter 8: PCB Design

Introduction

PCB stands for Printed Circuit Board which is a board made from layers of electric conductors, metal interconnects, insulators, and other components such as diodes and resistors connected according to a design using copper tracks etched into the surface of the board.

Printed circuit boards (PCBs) are the foundational building block of most modern electronic devices. It can be simple single or more insulating or copper layered boards that contain the signal traces and the powers and grounds, the design of the layout of printed circuit boards can be as demanding as the design of the electrical circuit. PCBs are the foundation on which all the other electronic components are assembled onto. Semiconductors, connectors, resistors, diodes, capacitors, and all components are talking to one another through the PCB to make sure they function properly. and have created faster more reliable, and more cost-effective integrated circuits (ICs).

PCB Boards are made from materials like copper and fiberglass with components that are then soldered onto the board. The PCB board is primarily used in the electrical engineering field to create electronic circuits. They are typically made of copper and can be customized with different components for a wide range of purposes.

A PCB is a board with electronic circuits on it that connects electronic devices. PCBs can be found in all sorts of equipment including televisions, computers, mobile phones, laptops, and cars.

Software tools

EasyEDA software is a very good tool for PCB design. You can use the inbuilt simulator to test your designs. You can verify analog, digital, and mixed-signal circuits with spice subcircuits and models.

This free tool **gives many features as**:

- **Open-source designs.**
- **You can make a simulation.**
- Single platform for all the PCB manufacturing processes.
- Free and Low-Cost subscription.
- Automatic Libraries Upgrade.

On this online software program, we can make it free by using a cloud server for the Schematic, PCB layout, and PCB design 3D for any circuit or any system so we use it.

Schematic Designing

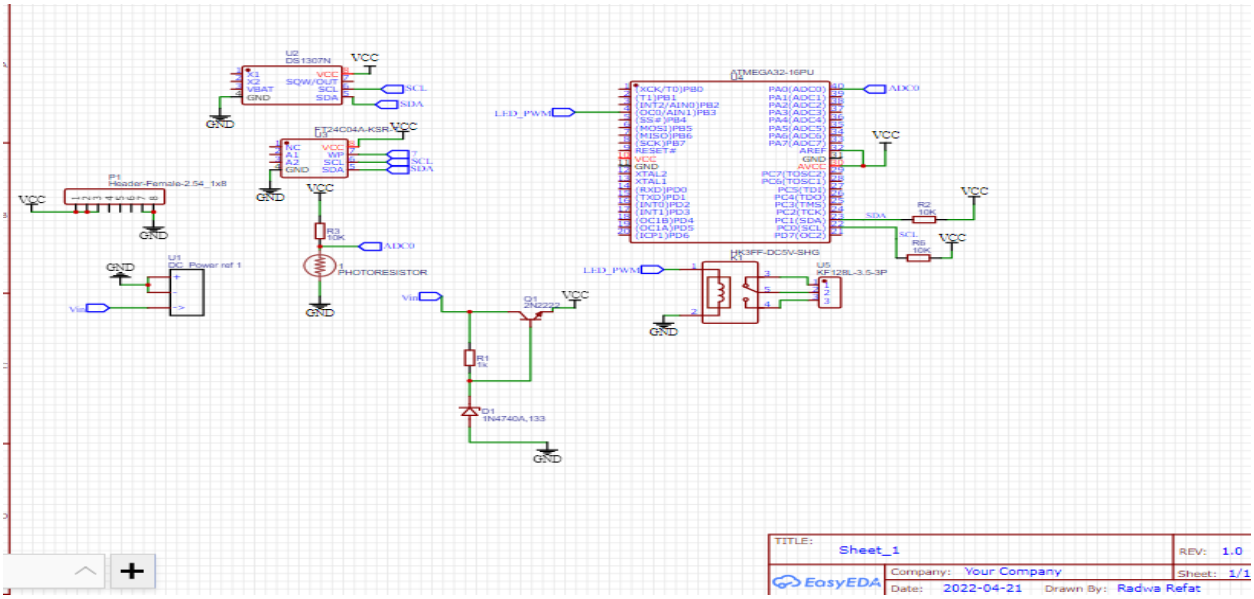
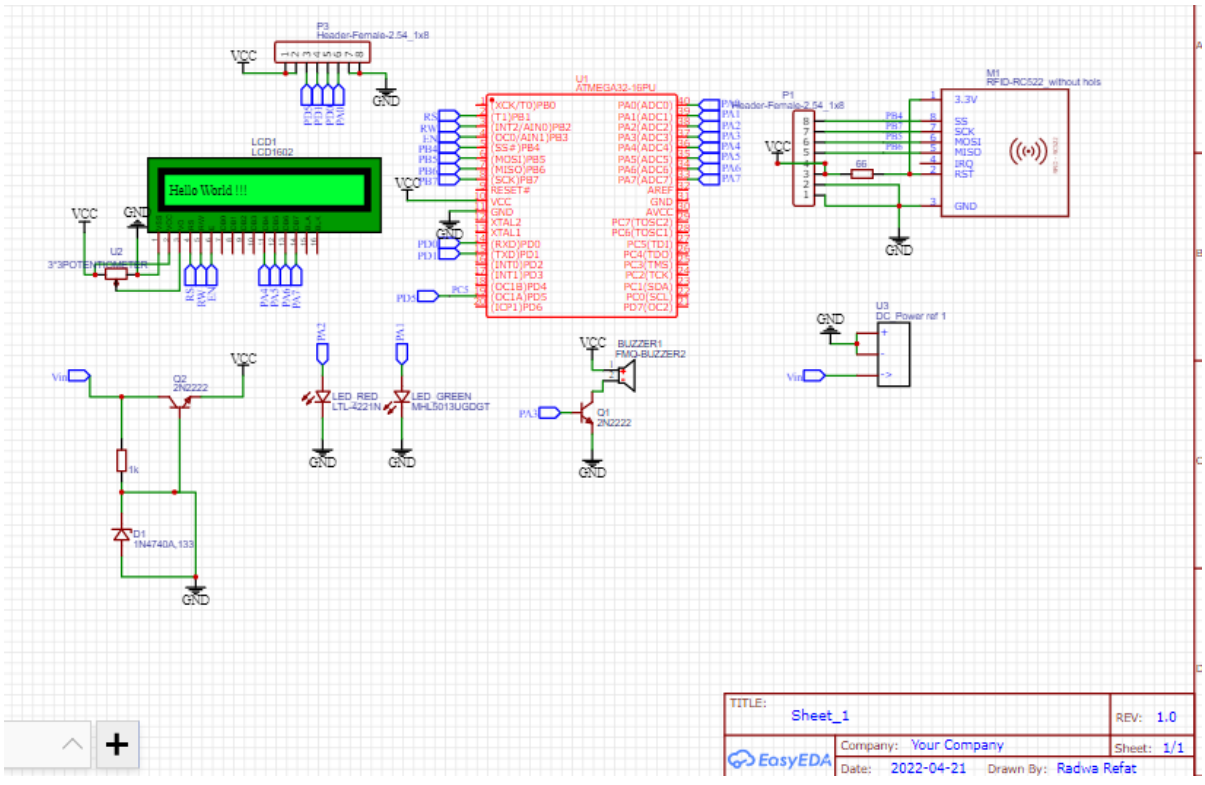


Figure 8-1“the Schematic for smart lighting system”.



\

Figure 8-2 “Gate Schematic “.

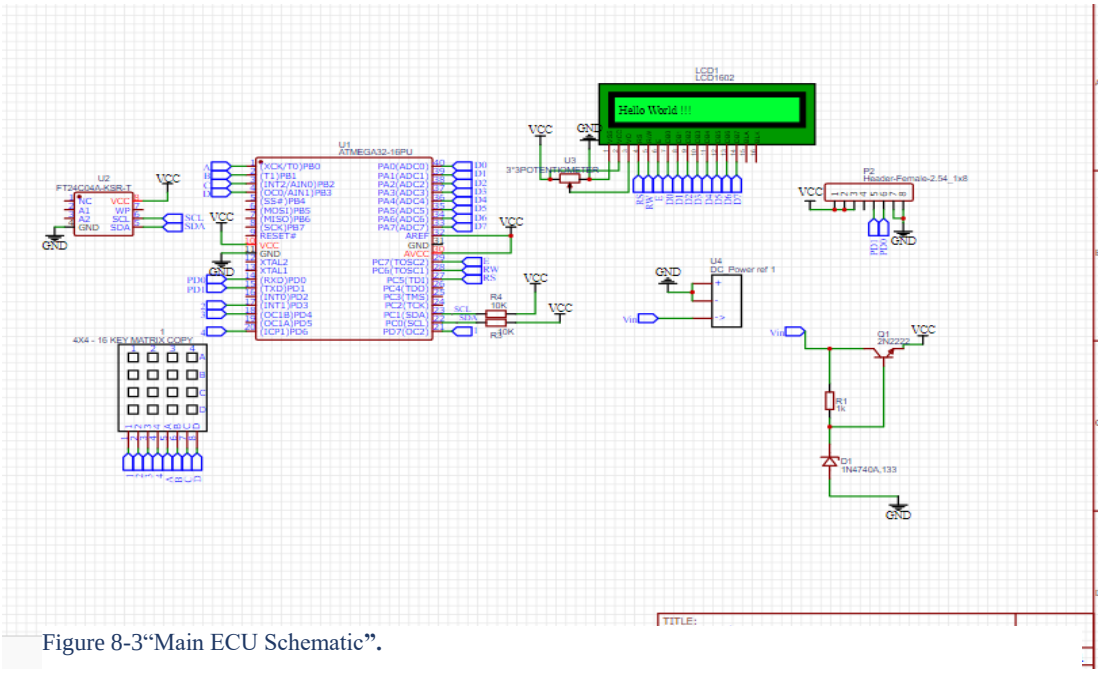


Figure 8-3“Main ECU Schematic”.

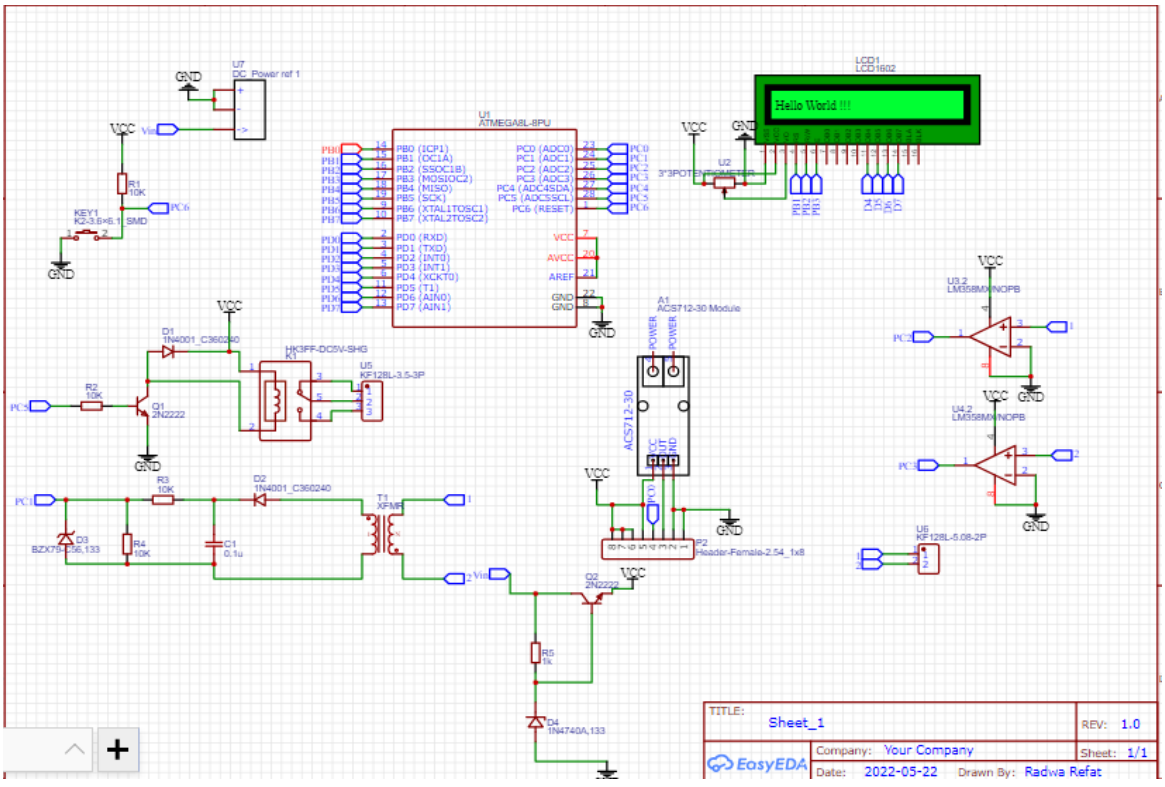


Figure 8-4“Smart Metering Schematic System “.

PCB Layout

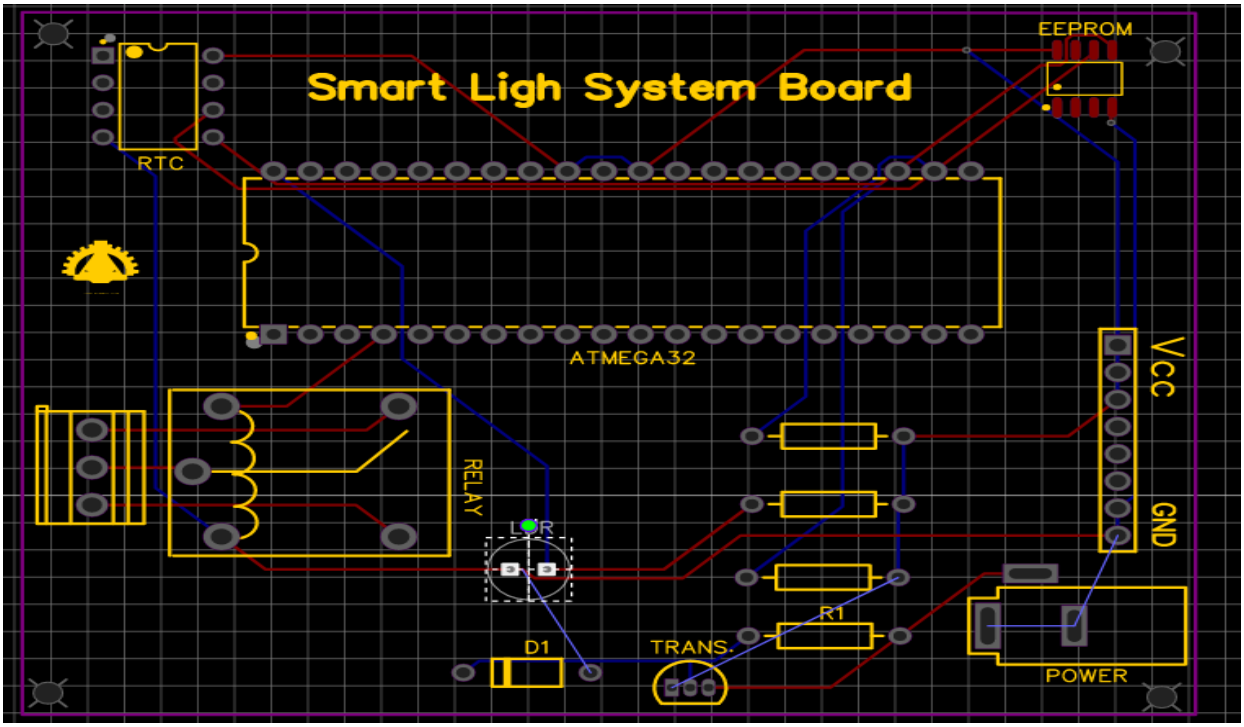


Figure 8-5"PCB layout for smart light system".

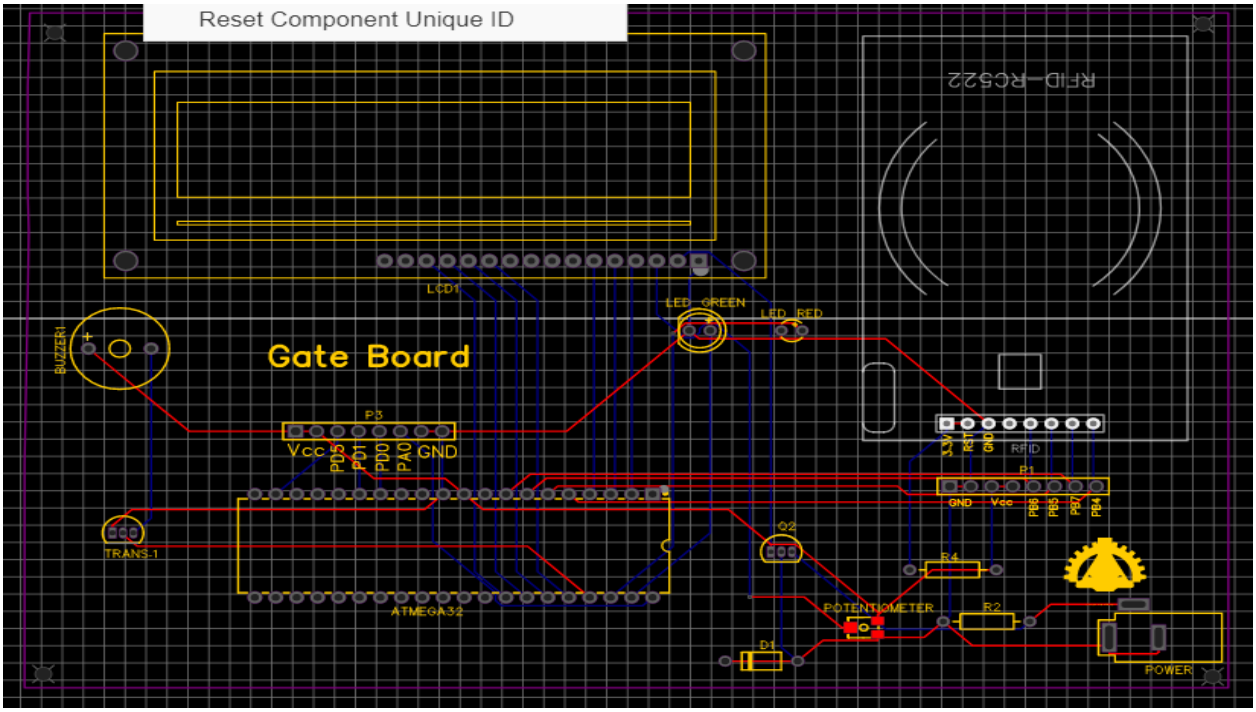


Figure 8-6"Gate PCB layout".

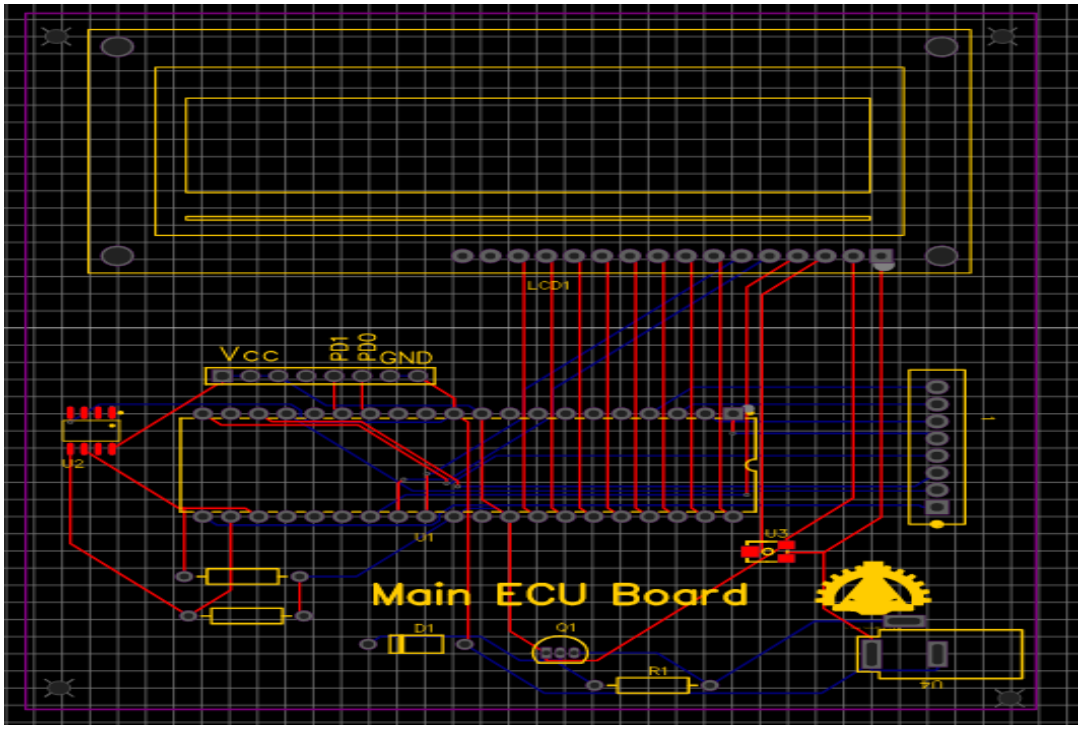


Figure 8-7"Gate PCB layout".

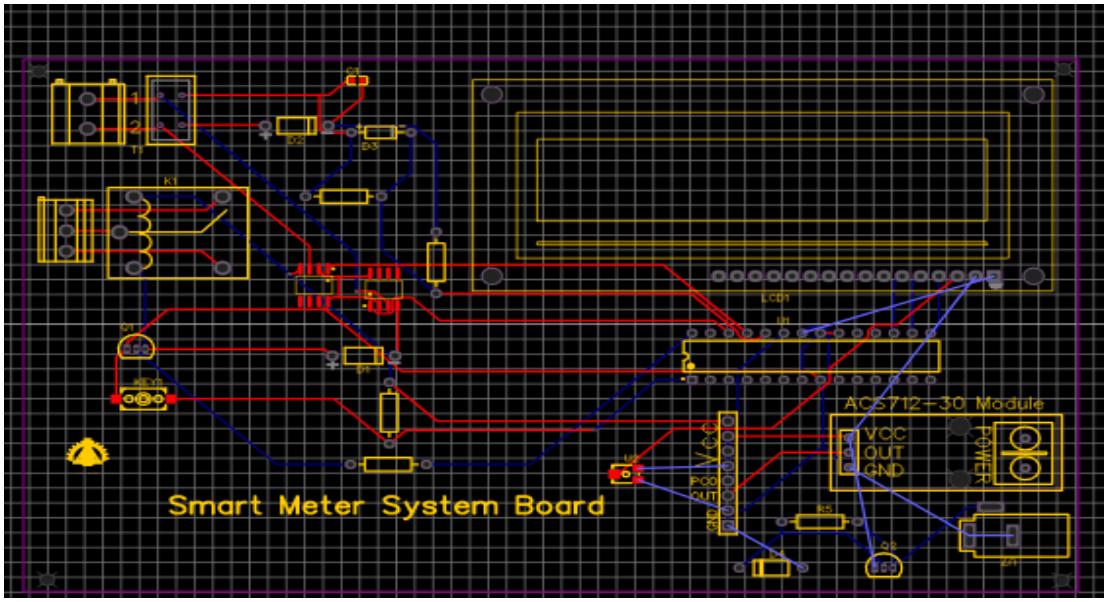


Figure 8-8 Smart Metering PCB layout".

PCB 3D View

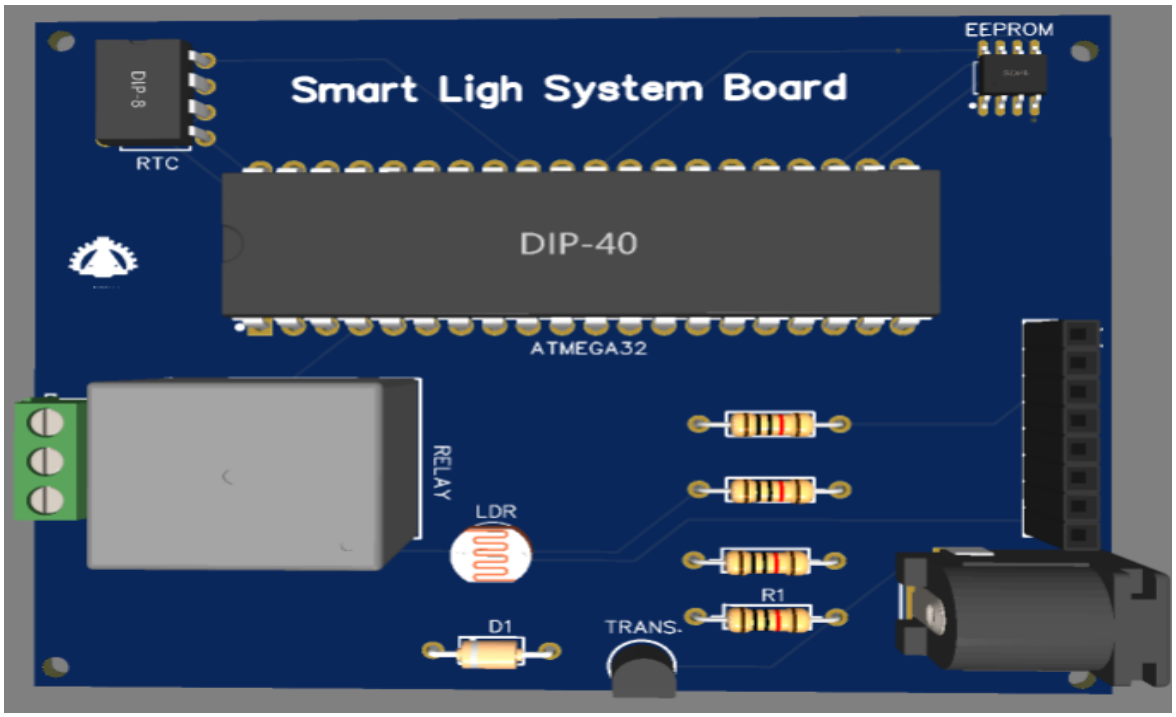


Figure 8-9“Smart Light 3D”.

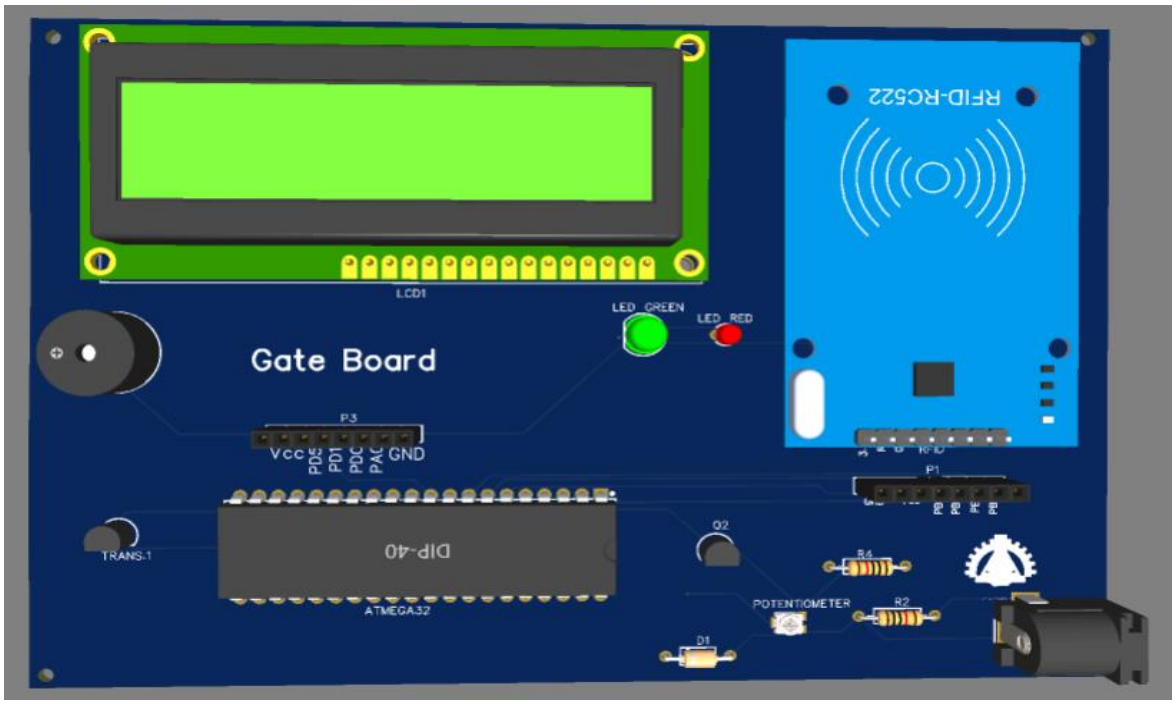


Figure 8-10 “Gate Board 3D”.

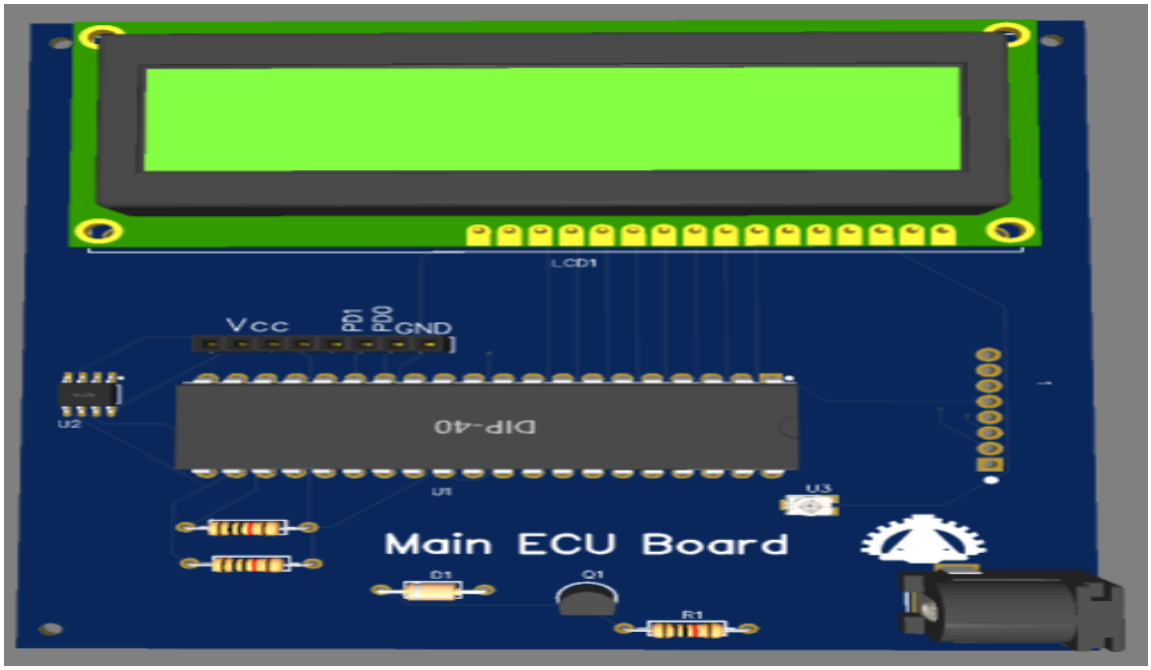


Figure 8-11“Main ECU Board 3D”.

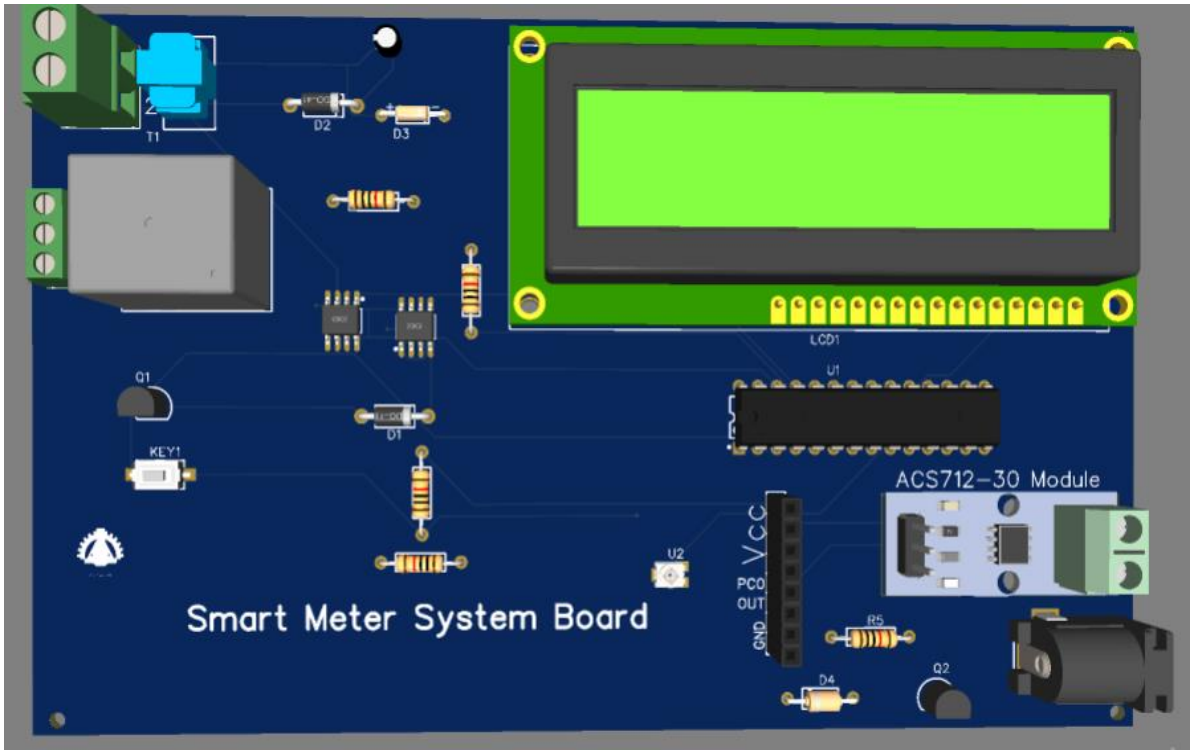


Figure 8-12 “Smart Meter Board 3D “.

Chapter 9: IOT & Cloud

Atmega32 microcontroller is a low-cost 8-bit microcontroller and comes with more number of GPIO's than its previous version of microcontrollers. It has all the commonly used communication protocols like UART, USART, SPI, and I2C. It has wide applications in robotics, automobile, and automation industries because of its wide community support and simplicity.

But Atmega32 doesn't support any of the wireless communication protocols such as Wi-Fi and Bluetooth which limits its application areas in a domain like IoT. To overcome this limitation other controllers can be interfaced which has wireless protocols. There are a number of controllers which supports wireless protocols like the widely used ESP8266,

We will interface Atmega32 with ESP8266 NodeMCU to make it communicate wirelessly through the internet. ESP8266 NodeMCU is a widely used Wi-Fi module with community support and easily available libraries. Also, ESP8266 NodeMCU is easily programmable with Arduino IDE. ESP8266 can be interfaced with any microcontroller.

1. Objective:

The target of interfacing NodeMCU with the atmega32 microcontroller is to collect the data from our systems: smart parking, smart metering, and smart lighting and send it to the cloud through the Wi-Fi module esp8266 to be stored, analyzed, and displayed (Visualized). The IoT and cloud interface with our three local systems will extend the systems to share, store and visualize these data.

2. Tools and Software:

2.1. NodeMCU

NodeMCU is an open-source LUA-based firmware developed for the ESP8266 Wi-Fi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board.

LUA is a powerful and fast programming language that is easy to learn and use and to embed into your application. LUA is designed to be a lightweight embeddable scripting language. It is used for all sorts of applications, from games to web applications and image processing.

NodeMCU Dev Kit/board consists of ESP8266 wifi-enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. It supports serial communication protocols i.e. UART, SPI, I2C, etc.

2.2. ESP8266

ESP8266 is Wi-Fi enabled system-on-chip (SoC) module developed by the Espressif system. It is mostly used for the development of IoT (Internet of Things) embedded applications.

The ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability with the advantage of good speed and processing power. ESP8266 module can be programmed in two ways: By LUA scripting and the second way is by Arduino programming.

The ESP8266 Wi-Fi Module is a self-contained SOC with an integrated TCP/IP protocol stack that can give any microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions.

"Fig 9.1 ESP8266"

ESP8266 comes with capabilities of

- 2.4 GHz Wi-Fi (802.11 b/g/n, supporting WPA/WPA2),
- general-purpose input/output (16 GPIO),
- Inter-Integrated Circuit (I^2C) serial communication protocol,
- analog-to-digital conversion (10-bit ADC)
- Serial Peripheral Interface (SPI) serial communication protocol,
- I^2S (Inter-IC Sound) interfaces with DMA(Direct Memory Access) (sharing pins with GPIO),
- UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and
- Pulse-width modulation (PWM).

It employs a 32-bit RISC CPU based on the Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz). It has a 64 KB boot ROM, 64 KB instruction RAM and 96 KB data RAM. External flash memory can be accessed through SPI.

ESP8266 module is low cost standalone wireless transceiver that can be used for end-point IoT developments.

To communicate with the ESP8266 module, microcontroller needs to use set of AT commands. Microcontroller communicates with ESP8266-01 module using UART having specified Baud rate.

There are many third-party manufacturers that produce different modules based on this chip. So, the module comes with different pin availability options like,

- ESP-01 comes with 8 pins (2 GPIO pins) – PCB trace antenna. (shown in above figure)
- ESP-02 comes with 8 pins, (3 GPIO pins) – U-FL antenna connector.
- ESP-03 comes with 14 pins, (7 GPIO pins) – Ceramic antenna.
- ESP-04 comes with 14 pins, (7 GPIO pins) – No ant.

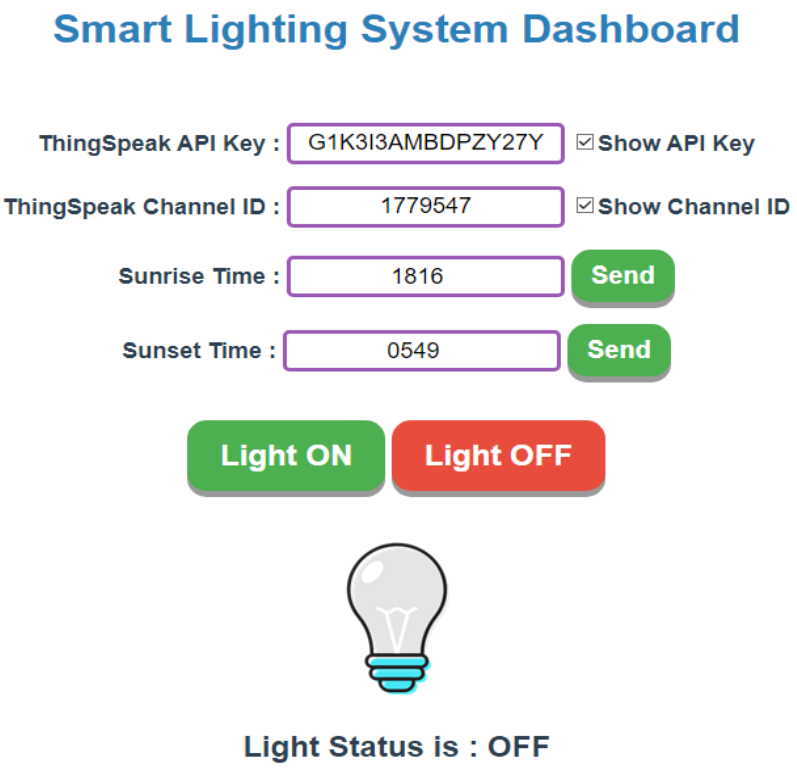
2.3. ThingSpeak Paltform:

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak from your devices, create instant visualizations of live data, and send alerts using web services. ThingSpeak provides a platform to quickly collect and analyze data from internet connected sensors. ThingSpeak includes a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications. It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists) But it should work with all kind of Programming Languages, since it uses a REST API and HTTP. At the heart of ThingSpeak is a time-series database. ThingSpeak provides users with free time-series data storage in channels. Each channel can include up to eight data fields.

2.4. Web page:

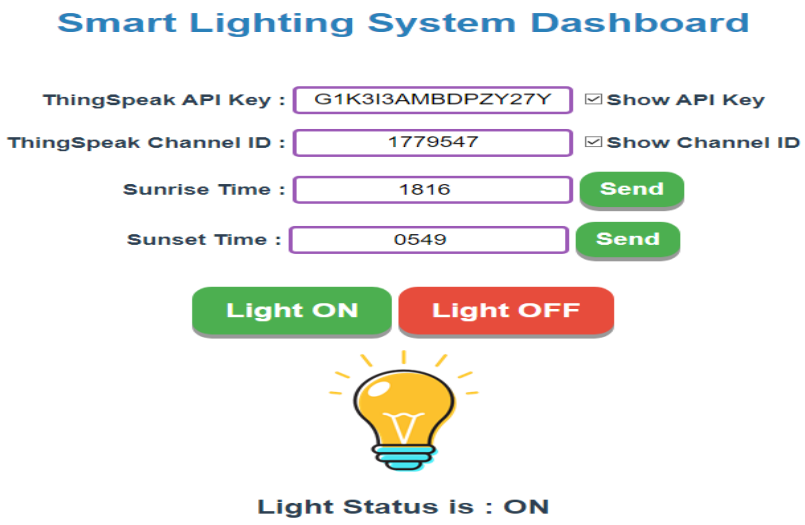
We used an html to create a web page to display and visualize the collected data and CSS to beautify the web page and json parse to get and post data to and from the cloud and here is what our pages look like under test of the system.

2.4.1 Smart lightening system with load off



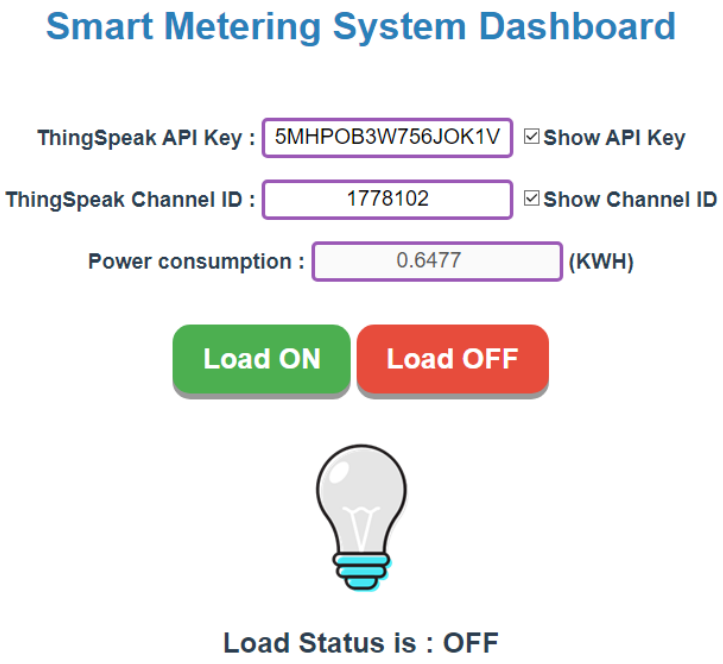
"Fig 9.2 SLS with load off"

2.4.2 Smart lightening system with load on



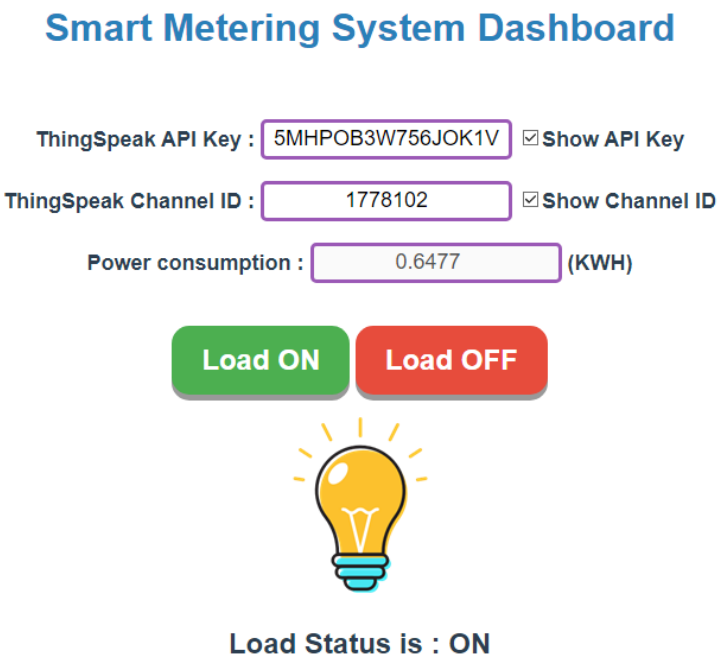
"Fig 9.3 SLS with load on".

2.4.3 Smart metering system with load off:



"Fig 9.4 SMS with load off"

2.4.4 Smart metering system with load on:



"Fig 9.5 SMS with load on"

2.4.5 Smart parking system

Smart Parking System Dashboard

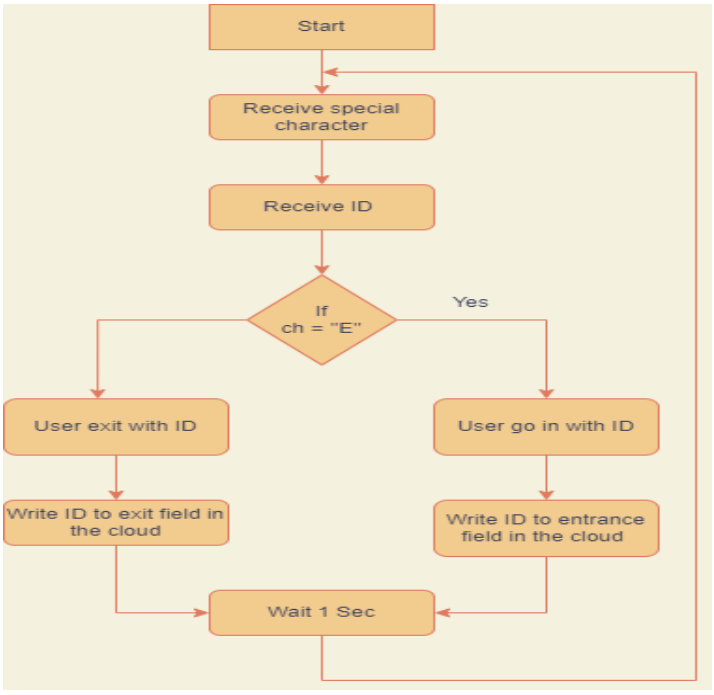
USER	ID	Time	Status
User 103	103	2022-07-11T22:05:00Z	Exist
User 112	112	2022-07-11T22:49:45Z	Entered
User 1	1	2022-07-11T22:50:04Z	Entered
User 100	100	2022-07-11T22:50:22Z	Entered
User 1	1	2022-07-11T22:50:51Z	Exist
User 112	112	2022-07-11T22:51:41Z	Exist

"Fig 9.6 SPS "

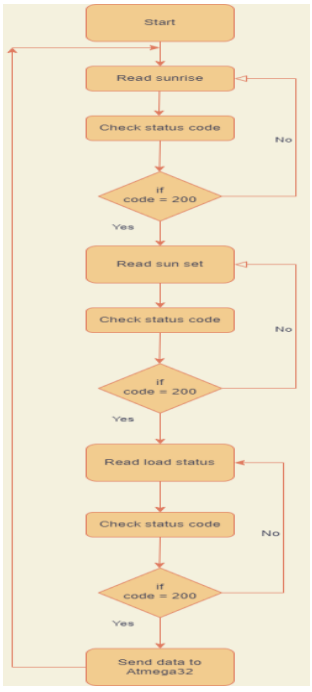
3. Cloudinterface with the Systems:

After every system collects its own data it start communicating with the cloud in our case is the things speak cloud in which we create a channel for each system to store its collected data and display it. The workflow of each system is as flow chart shows

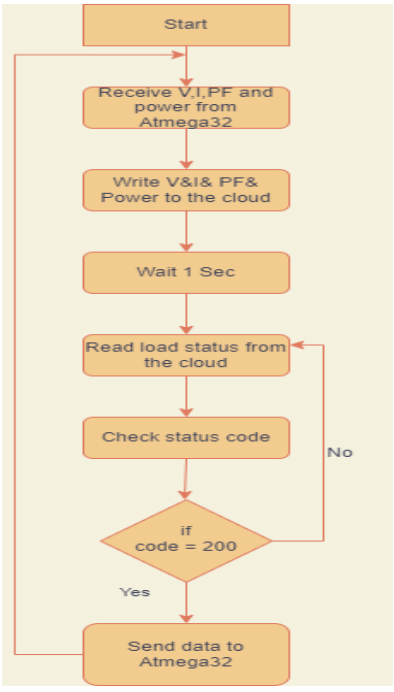
3.1. Flow charts



"Figure 9.7 SPS flow chart"



"Fig 9.8 SLS flow chart"



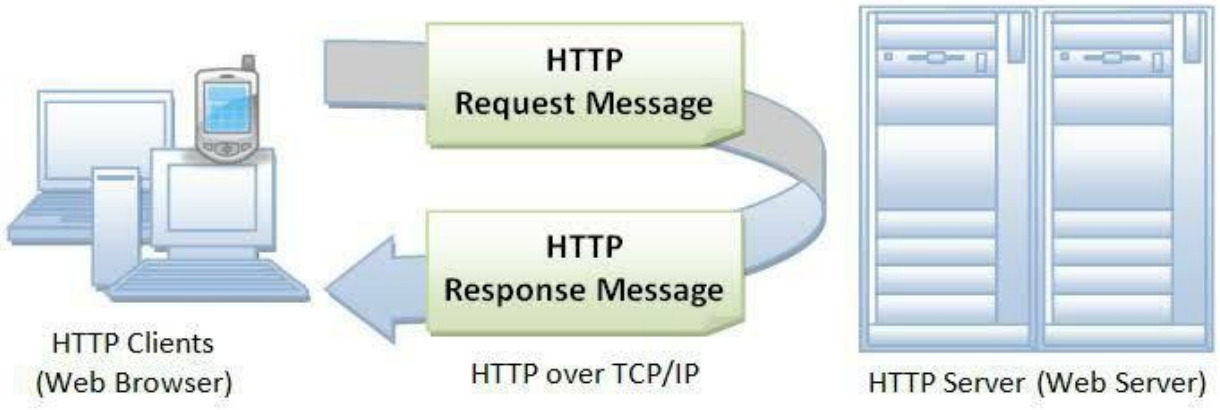
"Fig 9.9 "SMS flow chart"

4. Cloud collected data:

On things speak cloud we have created three main channels one for each system.

- Smart metering channel
- Smart lightening channel
- Smart parking channel

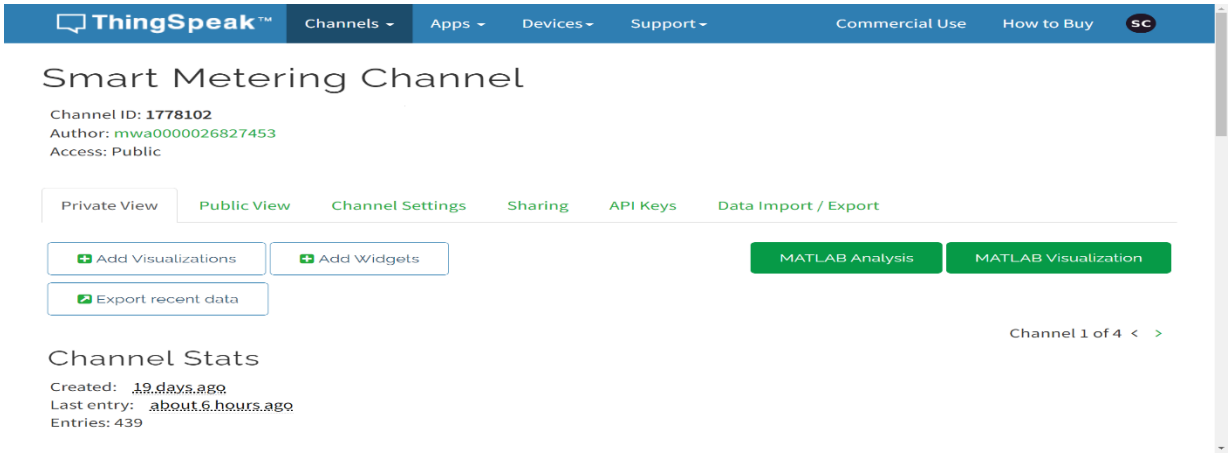
Sequence of collecting data from cloud: 1- node send http request it's type is GET. GET The GET method is used to retrieve information from the given server using a given URI Requests using GET should only retrieve data and should have no other effect on the data. 2- Server response a massage containing status of http request. If code status equal 200 that mean http request is done correctly without any error.



"Fig 9.10 HTTP protocol"

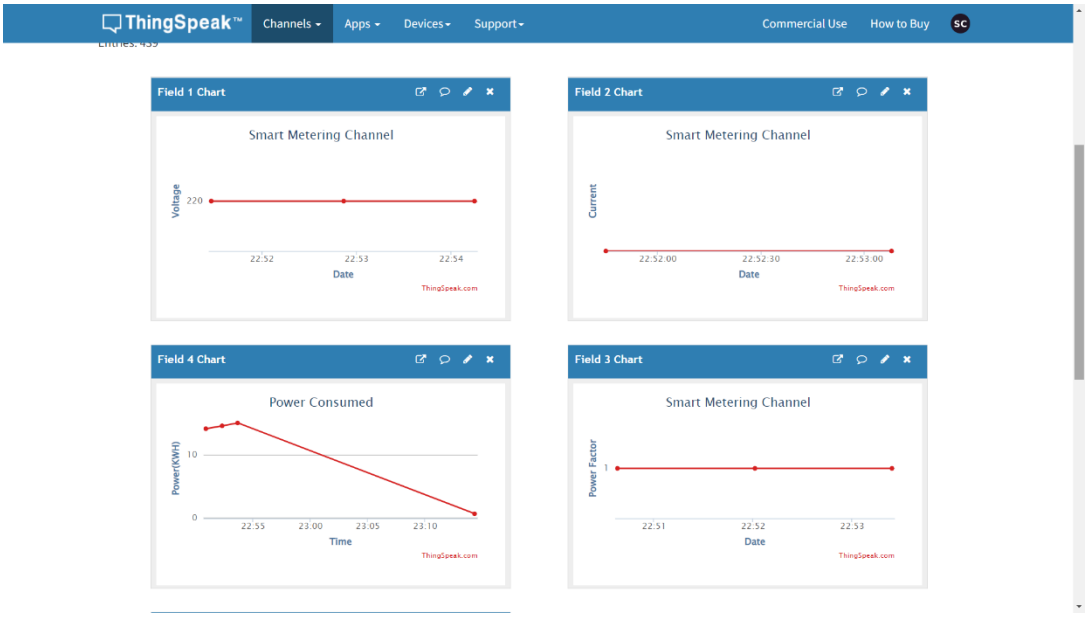
On each channel there are some fields for each collected variable data like voltage, current, power factor, and power in smart metering system. Here is the collected data from the cloud

4.1. Smart metering channel:



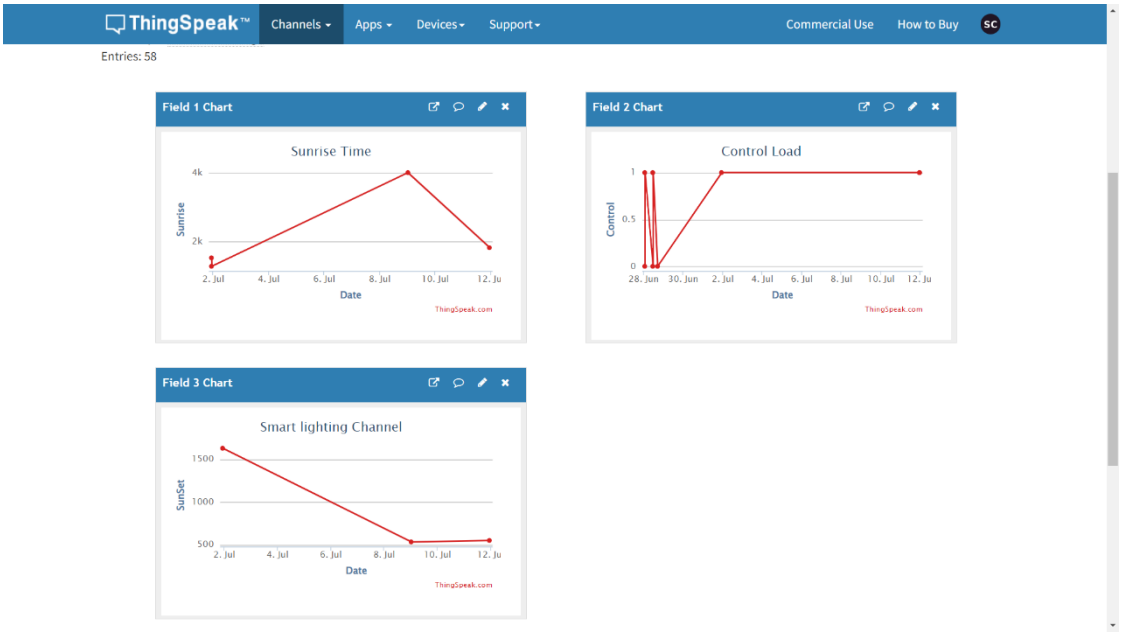
"Fig 9.11 SMS Channel"

4.1.1. Smart metering fields:



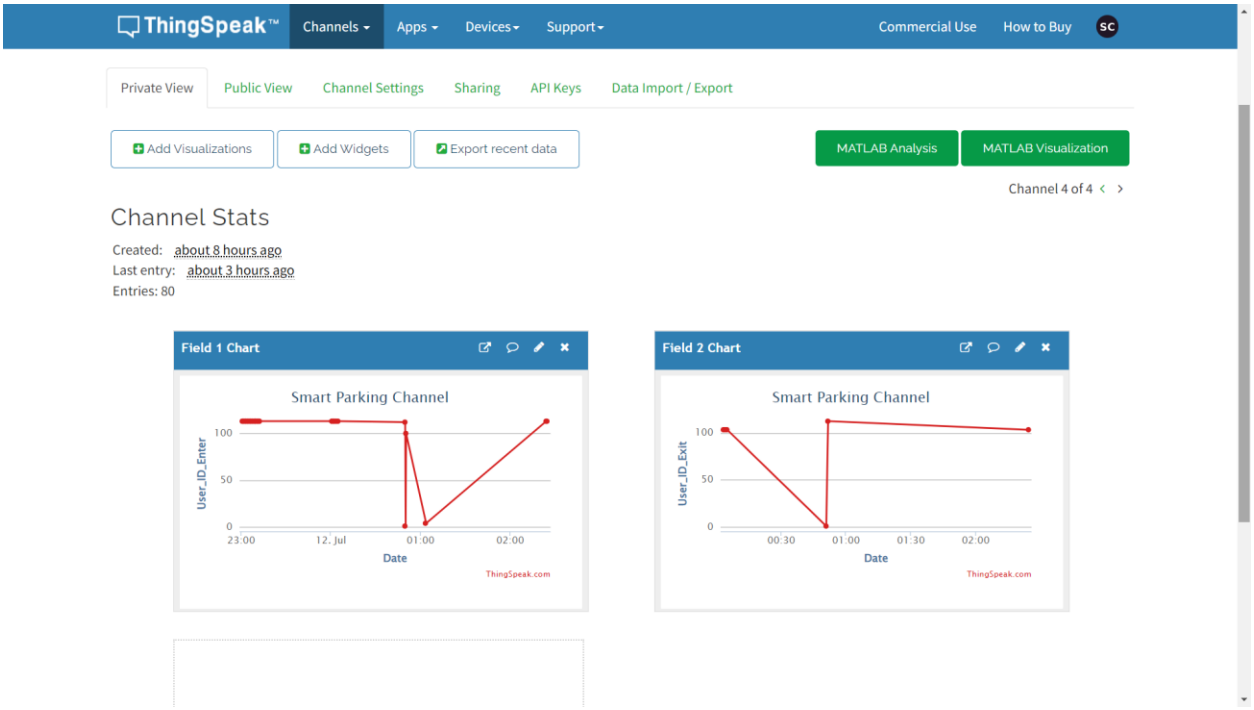
"Fig 9.11 SMS fields"

4.2. Smart lightening fields



"Fig 9.11 SLS fields"

4.3 **Smart parking fields**



"Fig 9.11 SPS fields"

References

- ATMEGA32 datasheet. (2022). Retrieved 12 July 2022, from https://datasheetspdf.com/pdf-file/219613/ATMELCorporation/ATMEGA32/1
- Preethi, V.Preethi & Gollaprolu, Harish. (2016). Design and implementation of smart energy meter. 1-5. 10.1109/INVENTIVE.2016.7823225.
- Rahim, B.A. (2012). Automated Wireless Meter Reading System for Monitoring and Controlling Power Consumption.
- Thoisana Singh, C. (2019). Cost Effective Power Factor Measurement Using Microcontroller ATmega8. from https://www.ijstr.org/final-print/july2019/Cost-Effective-Power-Factor-Measurement-Using-Microcontroller-Atmega8-.pdf
- ACS712 Datasheet. (2022). Retrieved 12 July 2022, from https://pdf1.alldatasheet.com/datasheet-pdf/view/428383/ALLEGRO/ACS712.html
- LM358 datasheet. (2022). Retrieved 12 July 2022, from http://www.unisonic.com.tw/datasheet/LM358.pdf
- Awodeyi, Afolabi & Samuel, Isaac & Akindele, Ayoola & Matthews, Victor. (2018). Design and Construction of a Microcontroller Based Automated Intelligent Street Lighting System. International Journal of Scientific and Engineering Research. 9.
- (2022). Retrieved 12 July 2022, from https://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter12.pdf
- (2022). Retrieved 12 July 2022, from https://www.researchgate.net/publication/299406927_PCB_Design_Process_and_Fabrication_Challenges
- What is a PCB and Why Do We Need Them - Custom Materials Inc. (2022). Retrieved 12 July 2022, from https://custommaterials.com/what-is-pcb/
- oEasyEDA : Circuit Design, Simulation, PCB Layout, PCB Ordering & Uses. (2022). Retrieved 12 July 2022, from https://www.watelectronics.com/easyeda-pcb-designing-circuit-simulation/
- (2022). Retrieved 12 July 2022, from https://pic-microcontroller.com/easyeda-features-for-schematic-and-pcb-design/
- D'Arcio, J. (2022). What is a Printed Circuit Board (PCB)? - Printed Circuits LLC. Retrieved 12 July 2022, from https://www.printedcircuits.com/what-is-a-pcb/
- F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.
- Elgendy, M., n.d. *Deep Learning for Vision Systems*.
- Chollet, F., 2022. *Deep Learning With Python, Second Edition*. Greenwich, USA: Manning Publications.
- Amer, Firas & Al-Tamimi, Mohammed. (2022). Face Mask Detection Methods and Techniques: A Review. The International Journal of Nonlinear Analysis and Applications (IJNAA). 13. 3811-3823. 10.22075/ijnaa.2022.6166.
- A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342585.
- Nazeer, Shahrin & Omar, Normah & Khalid, Marzuki. (2007). Face Recognition System using Artificial Neural Networks Approach. Proceedings of ICSCN 2007: International Conference on Signal Processing Communications and Networking. 420 - 425. 10.1109/ICSCN.2007.350774.
- L. Li, X. Mu, S. Li and H. Peng, "A Review of Face Recognition Technology," in IEEE Access, vol. 8, pp. 139110-139120, 2020, doi: 10.1109/ACCESS.2020.3011028.