**REPORT**

**1. INTRODUCTION**

1] ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

2] The ESP32 has a total of 34 digital pins. These pins are similar to Arduino digital pins which allows you to add LED, Resistors, sensors, buttons, etc. to our projects.

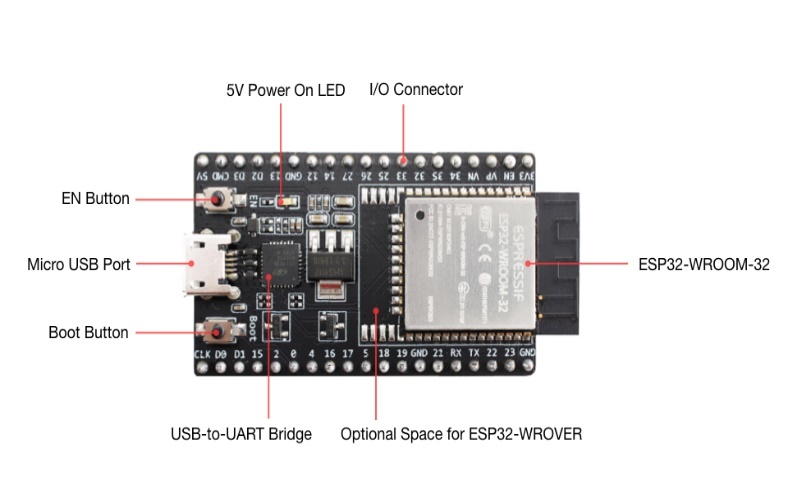
3] Most of these pins support the use of internal pull-up, pull-down, and high impedance status as well. This makes them ideal for connecting buttons and matrix keyboards, as well as for applying LED control techniques such as the well-known Charlieplexing.

4] ESP32 WROOM module has 25 GPIO pins out of which there are only input pins, pins with input pull up and pins without internal pullup.

5] Maximum current drawn per a single GPIO is 40mA according to the “Recommended Operating Conditions” section in the ESP32 datasheet.

6] ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling.

7] For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.



**Fig 1.1: ESP32**

**2. Peripherals and Sensors**

2.1 General Purpose Input/Output Interface (GPIO)

ESP32 has 34 GPIO pins which can be assigned various functions by programming the appropriate registers. There are several kinds of GPIOs: digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs.

2.2 Analog ­to ­Digital Converter (ADC)

ESP32 integrates two 12-bit SAR ADCs and supports measurements on 18 channels (analog-enabled pins). The ULP coprocessor in ESP32 is also designed to measure voltage, while operating in the sleep mode, which enables low-power consumption. The CPU can be woken up by a threshold setting and/or via other triggers.

2.3 Digital­ to­ Analog Converter (DAC)

Two 8-bit DAC channels can be used to convert two digital signals into two analog voltage signal outputs. The design structure is composed of integrated resistor strings and a buffer. This dual DAC supports power supply as input voltage reference. The two DAC channels can also support independent conversions.

3. **Calculations**

1] The parameters for the above normal waveform are:

width of 1st pulse

2nd pulse = a + 50μS = 700 + 50μS = 750μS

3rd pulse = a + 100μS = 700 + 100μS = 800μS

4th pulse = a + 150μS = 700 + 150μS = 850μS

2] The possible system modes are as follows-

c-3 pulse = 18-3pulse = 15 pulse

c+3 pulse = 18+3pulse = 21 pulse

Half d and b time until switch set back to 0

d = 0.25 and b = 1050

2] Calculation of parameters Each student will have a set of (a, b, c, d, and mode) parameters. To calculate your set of parameters, use the following information. Alphabet is numbered - a=1, b=2, c=3, … m=13 n=13, o=12, p=11, … z=1

Letters – GUNJAN

a = first letter \* 100μS = 700μS

b = second letter \* 100μS = 2100μS

c = third letter + 4=14+4 = 18μS

d = fourth letter \* 500μS = 500μS = 0.5ms

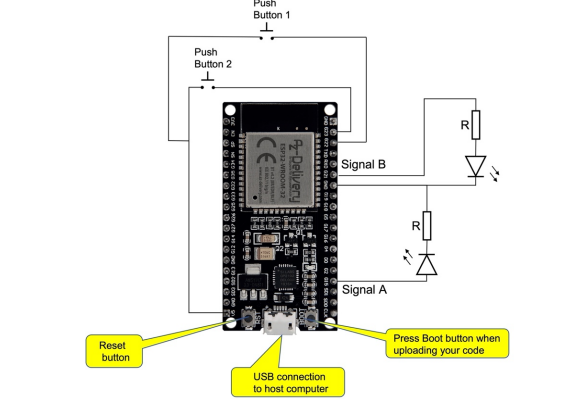
mode = remainder (fifth letter/4) +1 = (1/4) +1 = 1

Build the electronic circuit.

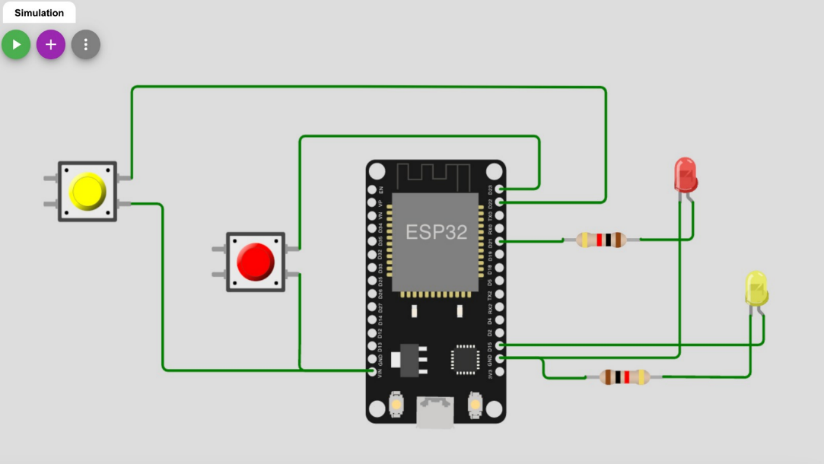
Build the circuit using the provided equipment (ESP32, 2 switches, 2 LEDs and an oscilloscope to visualize Signals A and B)

**4. ESP32 Circuit Diagram**

ESP32 circuit diagram simulates on WOKWI , Wokwi is an**online simulator for Arduino and Electronics**. It's designed for makers, by makers. You can use Wokwi to learn Arduino programming, prototype your ideas, and share your projects with other makers. Below fig extruded



**Fig: Circuit Diagram**



**Fig: Simulation of circuit diagram**

**Task 1**

**Code with Arduino:**

//Assignment1 ESP32

const int led\_1 = 15;

const int led\_2 = 21;

const int button\_1 = 22;

const int button\_2 = 23;

void setup() {

Serial.begin(115200);

pinMode(15, OUTPUT);

pinMode(21, OUTPUT);

pinMode(22, INPUT);

pinMode(23, INPUT);

}

void mod\_a(int pin, int puls\_a, int time\_inc, int pause\_b, int d) {

digitalWrite(led\_2, HIGH);

delayMicroseconds(50);

digitalWrite(led\_2, LOW);

delayMicroseconds(50);

for(int i = 0; i < 18; i ++) {

digitalWrite(pin, HIGH);

delayMicroseconds(puls\_a);

puls\_a = puls\_a + time\_inc;

Serial.println(puls\_a);

digitalWrite(pin, LOW);

delayMicroseconds(pause\_b);

Serial.println("NORMAL MODE");

}

digitalWrite(pin, LOW);

delayMicroseconds(d);

}

void mod\_b(int pin, int puls\_a, int time\_inc, int pause\_b, int d) {

digitalWrite(led\_2, HIGH);

delayMicroseconds(50);

digitalWrite(led\_2, LOW);

delayMicroseconds(50);

for(int i = 0; i < 15; i ++) {

digitalWrite(pin, HIGH);

delayMicroseconds(puls\_a);

puls\_a = puls\_a - time\_inc;

Serial.println(puls\_a);

digitalWrite(pin, LOW);

delayMicroseconds(pause\_b);

Serial.println("MODE 1");

}

digitalWrite(pin, LOW);

delayMicroseconds(d);

}

void loop() {

if(digitalRead(button\_1)==HIGH)

{

//mod\_a(led\_1,700, 50, 2100, 500);

//Serial.println(puls\_a);

digitalWrite(led\_1, LOW);

digitalWrite(led\_2, LOW);

}

else if(digitalRead(button\_2)==HIGH)

{

mod\_b(led\_1,700, 50, 2100, 500);

//Serial.println(puls\_a);

}

else

{

mod\_a(led\_1,700, 50, 2100, 500);

//Serial.println(puls\_a);

//digitalWrite(led\_1, LOW);

//digitalWrite(led\_2, LOW);

}

}

**Task 2**

Diagram

Description automatically generated

**Fig: Actual Circuit Diagram**

A screenshot of a computer

Description automatically generated with medium confidence

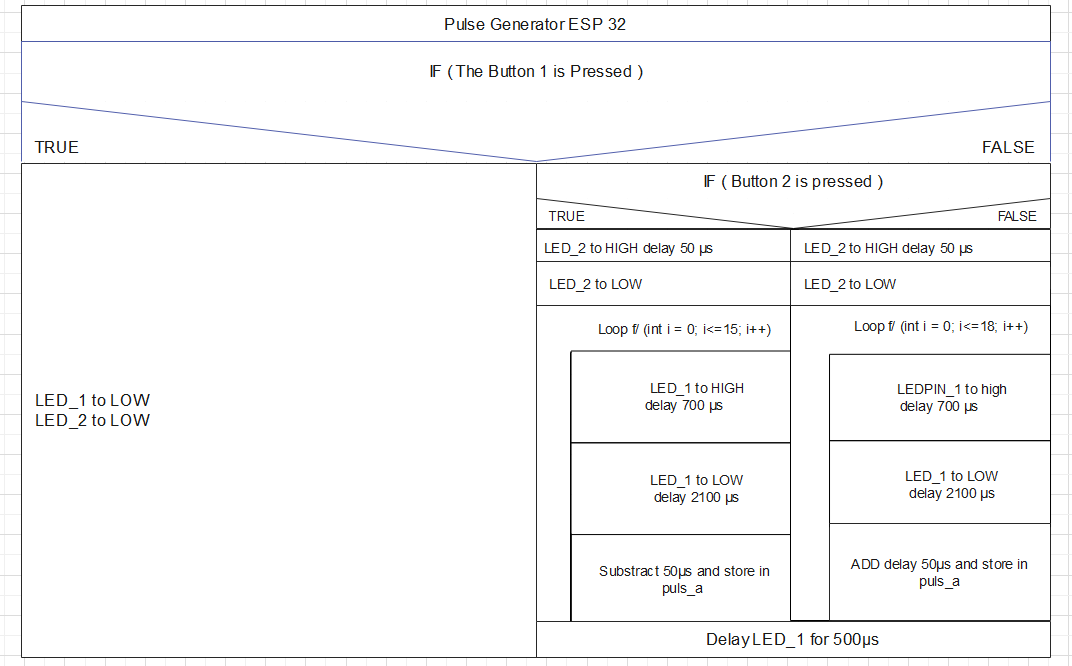
**Fig A: Pulse in Normal Mode**



**Fig B: Pulse in mode 1**

**Task 3**

**Nassi Shneiderman Diagram**



**Fig: N – S Diagram**

**Github Link :**

[gunjandod/ESP32-signal-regerator: Generating signal for 2 pulse (github.com)](https://github.com/gunjandod/ESP32-signal-regerator)