Electrical, Electronic and Computer Engineering
School of Engineering and Physical Sciences

Evrard EMONOT
H00385163

B31DG : Embedded Software

Assignment 1.

Summary

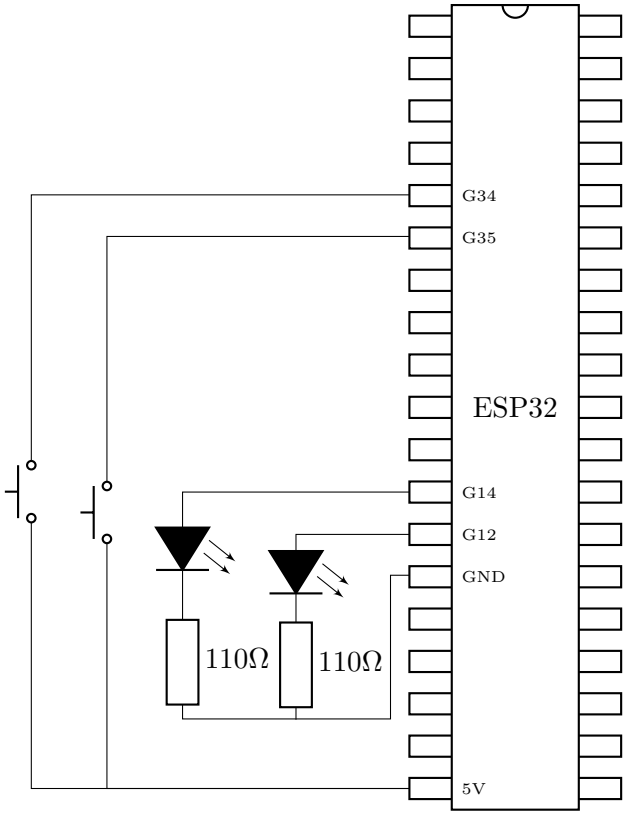
1 Hardware

1.1 Schematics

We are working on an ESP32

Material used

- ESP 32
- 2 Resistor of 230Ω
- 2 Green LED
- 2 switchs



1.2 Photos

Photos of the electronic assembly

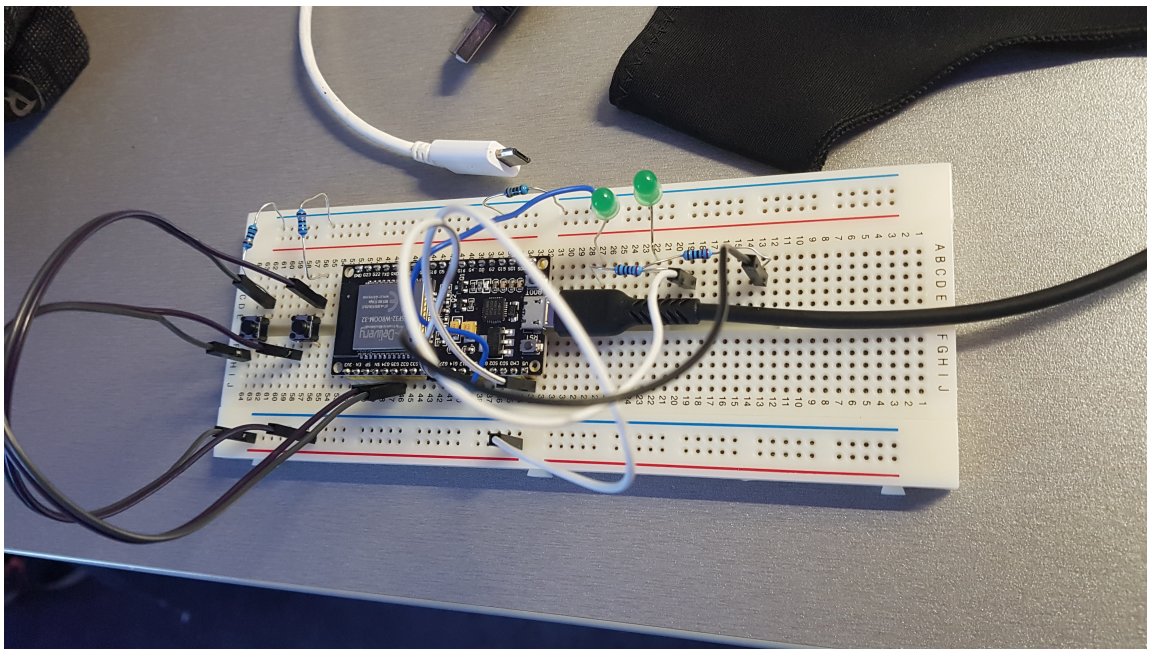


FIGURE 1 – Signal A off

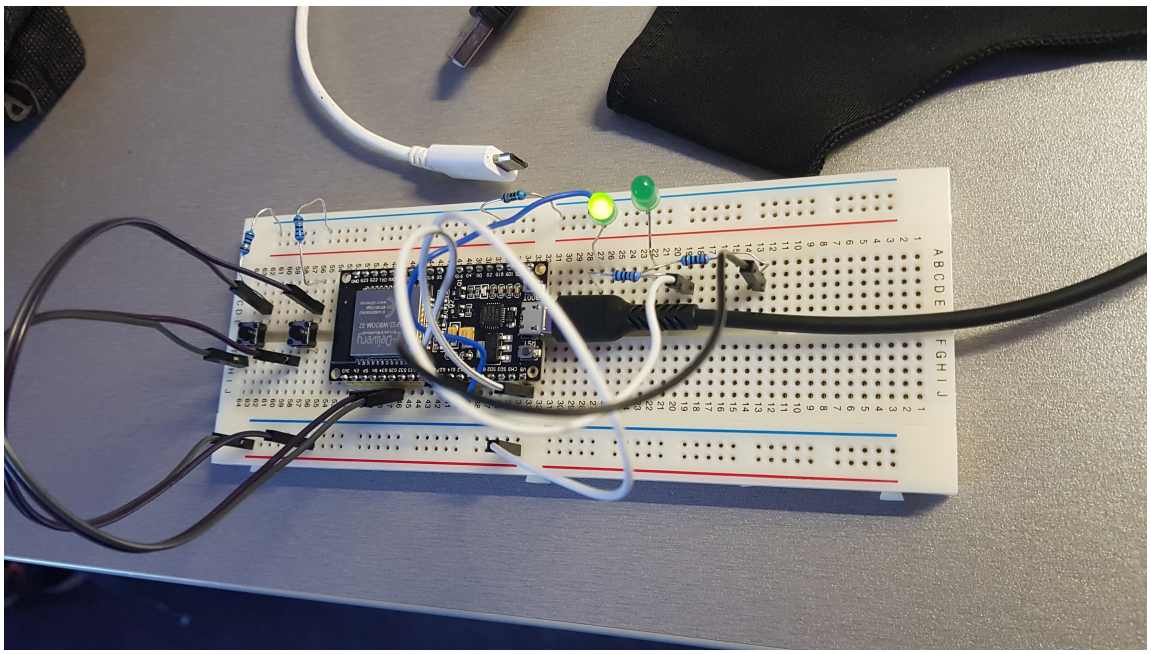


FIGURE 2 – Signal A on

2 Software

2.1 Delay parameters

In this section, We are going to calculate the delays of the signal using my name "emonot":

$$a = e \times 100 = 500\mu S$$

$$b = m \times 100 = 1300\mu S$$

$$c = o + 4 = 16 pulses$$

$$d = n \times 500 = 6500\mu S$$

$$B = 50ms$$

$$Mode = rem(\frac{o}{4}) + 1 = 1$$

If we pressed the second button, we apply the mode one which remove 3 pulses from c. So :

$$c = 13$$

2.2 Code commentary

This section describe the general structure of the code. This structure is divided into 4 sections.

sections

- Pin assignment
- Constant delay for the signal
- Pin property
- Algorithm loop

Section for the pin assignment :

```
// defining constant for the INPUT and OUTPUT of the ESP32
#define SIGNAL_A 14
#define SIGNAL_B 12
#define INPUT_1 35
#define INPUT_2 34
```

Some example of the constant declaration

```
int input_2_Btn = 0;
int const B = 50*1000;
int const b = 1300*1000;
```

Section for assigning pin property

```
void setup() {
  // initialize digital pins as inputs and outputs
  pinMode(SIGNAL_A, OUTPUT);
  pinMode(SIGNAL_B, OUTPUT);
  pinMode(INPUT_1, INPUT);
  pinMode(INPUT_2, INPUT);
}
```

This is an example of a section of the algorithm that read the states of the buttons and verify a condition. If the condition is met, it executes a certain number of pulses with the delay provided by the calculus in chapter 2.1.

```
  input_1_Btn = digitalRead(INPUT_1);
  input_2_Btn = digitalRead(INPUT_2);

  //4 conditions to verify the states of the buttons as there are pressed or not
  // if both button are not pressed, we run in normal mode

  if(input_1_Btn == LOW and input_2_Btn == LOW)
  {
    //we apply the bliking of the LEDS as requested in the assignement
    digitalWrite(SIGNAL_B, HIGH);
    delayMicroseconds(B);
    digitalWrite(SIGNAL_B, LOW);

    //we have 16 blink with a period increment between each blink
    for(i=0;i<16;i++){
      digitalWrite(SIGNAL_A, HIGH);
      delayMicroseconds(a);
      digitalWrite(SIGNAL_A, LOW);
      a += 50000;
      delayMicroseconds(b);
    }
    delayMicroseconds(d);
    digitalWrite(SIGNAL_B, LOW);
    a = 500*1000;
  }
}
```

The whole code is published on git, the link can be find in the README file

2.3 Nassi Shneiderman diagram

This diagram follows the code structure

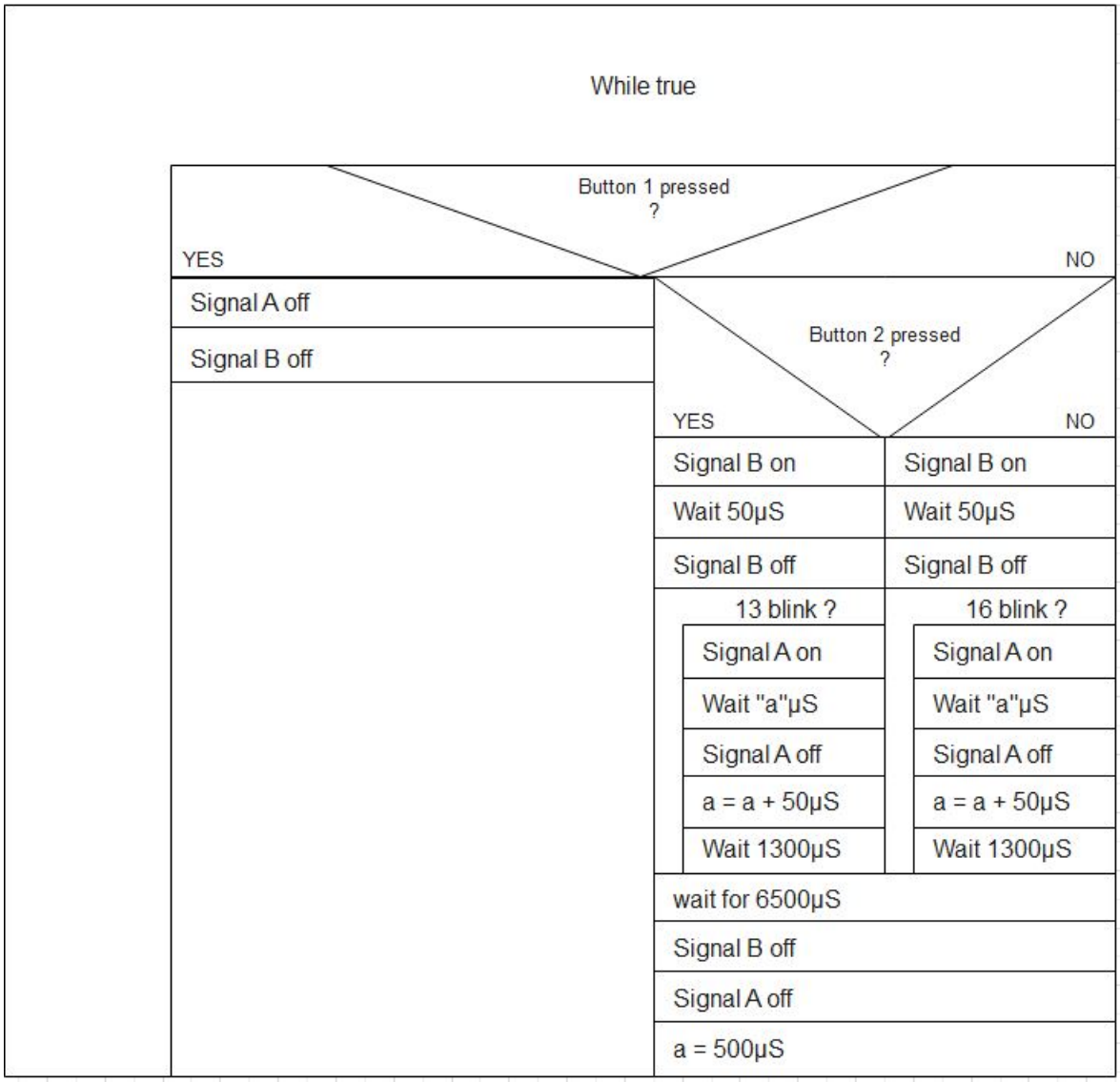


FIGURE 3 – Nassi Shneiderman diagram

2.4 Results

The code provided on git contains 2 files because with have 2 ways to verify our program.
First we can increase the delay of the constant to see if the algorithm is working properly.
If we run the normal code with delays in microseconds, then we verify some information on the oscilloscope.

In this case we can see that for each new 16 pulse of signal A small signal B is detected.

FIGURE 4 – The line in yellow is SIGNAL A, The line in blue is SIGNAL B

On this second image we can see that after that the pulse's delay of the A signal is bigger that the next ones which mean that the delay "a" is correctly increasing.

FIGURE 5 – The line in yellow is SIGNAL A, The line in blue is SIGNAL B