Embedded Software

Assignment 2

Sylvain Jannin

H00387879

Summary

Table of figures

1. Context and electronic design

The goal of this assignment is to implement a cyclic executive. There are 9 tasks, each task is easy to implement, and they are all executing at a different frequency.

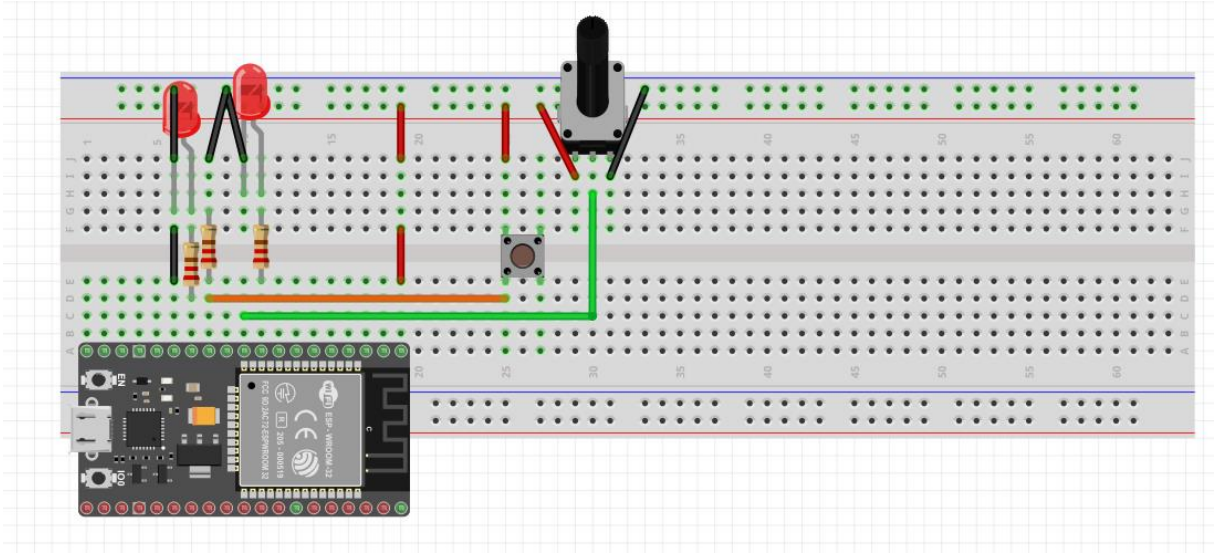

Figure 1 : electric schematic.

On the figure above there is the electric schematic. For all inputs, there are pull-down resistors in order to have better signals. There are also an LED and a resistor for task 1 to measure the signal.

On the figure below, the electronic assembly.

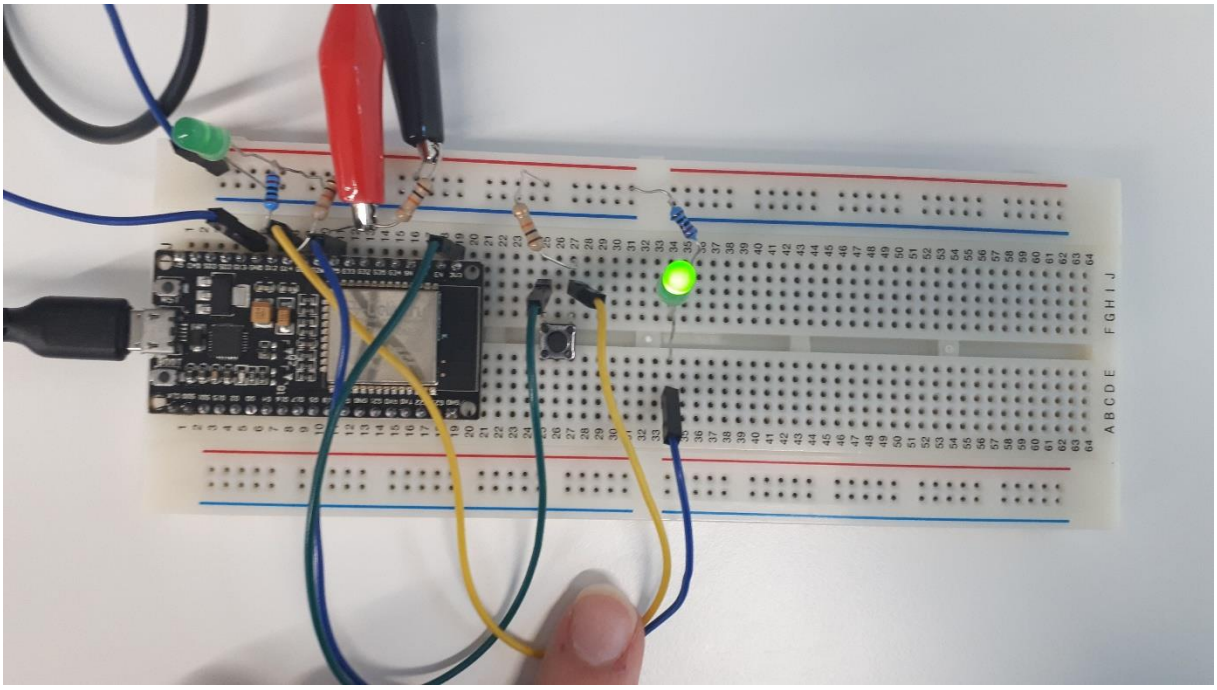

Figure 2 : electric assembly pictures.

2. Programming

2.1. Cycle executive

There are 9 task to organise within a cycle executive. The description of each task is the next part. On the below figure There is a schematic of our cycle executive for the first 100ms. There is a .xlsx with the whole cycle executive. This schematic shows the order of the called task. It does not represent the time between two calls.

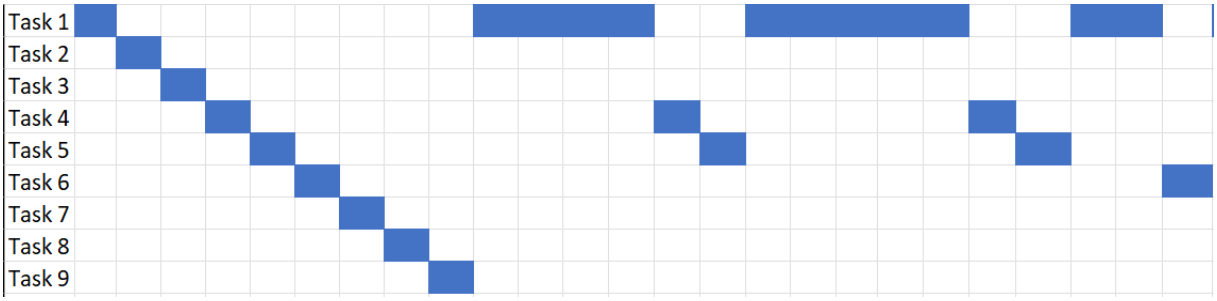

Figure 3: Cycle executive for the first 100ms.

To do the cycle executive we use a Ticker object from the Ticker library. The ticker calls a function at a defined rate that we set. In our case we call the function every 1ms. The idea is to count the number of times the function is called with a counter. This counter is a global variable that increments every time the function is called.

```
myTicker.attach_ms(1, my_function);
```

Figure 4: Setup of the Ticker for the cycle executive.

The function used as a cyclic executive is simply a list of if. Each if is used to check if the task must be called or not. For example, the task 3 has to be called every second. Since the function is called every millisecond, the counter increments every millisecond. So, every 1000ms we can call task3. To check if it has been 1000 times we simply do a modulo operation on the counter and see if the result is 0 or not: *if(Counter % TimeTask3 == 0)*.

```
void my_function()
{
  if(Counter % TimeTask1 == 0)
    task1();
  if(Counter % TimeTask2 == 0)
    task2();
  Counter++;
}
```

Figure 5: example of the cycle executive in Arduino.

2.2. Tasks

All tasks are easy to implement, and they are call in the function at the good rate (T is in ms).

- **Task 1**:
  - T = 9,250 ms rounded at 9ms (based on signal b of assignment 1)
  - Send a signal high for 50 µs then low
  - Use delay function
- **Task 2:**
  - T = 200 ms
  - Check the state of an input
- **Task 3:**
  - T = 1000 ms
  - Compute the frequency on an input signal
  - Use an interrupt on rising edge to read the time at every rising edge
  - Use global variable to note the last to time the external signal rose
  - $f = \frac{10^6}{current_{time} - previous_{time}}$. The measure of time is in µs, so we need to multiply the frequency by 10^6 to have it in Hz.
- **Task 4:**
  - T = 42 ms (rounded)
  - Read an analogic input
- **Task 5:**
  - T = 42 ms (rounded)
  - Take the last 4 analogic inputs and do compute the average
- **Task 6:**
  - T = 100 ms
  - Send 1000 times the command "*__asm__ __volatile__ ("nop")*"
  - Use a for loop
- **Task 7:**
  - T = 333 ms (rounded)
  - Check if the average of the last 4 inputs is bigger than 1,65V
  - Change the value of *error_code* accordingly
- **Task 8:**
  - T = 333 ms (rounded)
  - Turn on or off a light regarding the value of *error_code*.
- **Task 9:**
  - T = 5000 ms
  - Print the task 2 input state, task 3 input frequency and task 5 average analogic input.

3. Testing

To test task 1, we did as in assignment one a check the signal on an oscilloscope. Our signal has a period of 9ms and a width of 50µs. It is verified with the captures in the next figures.

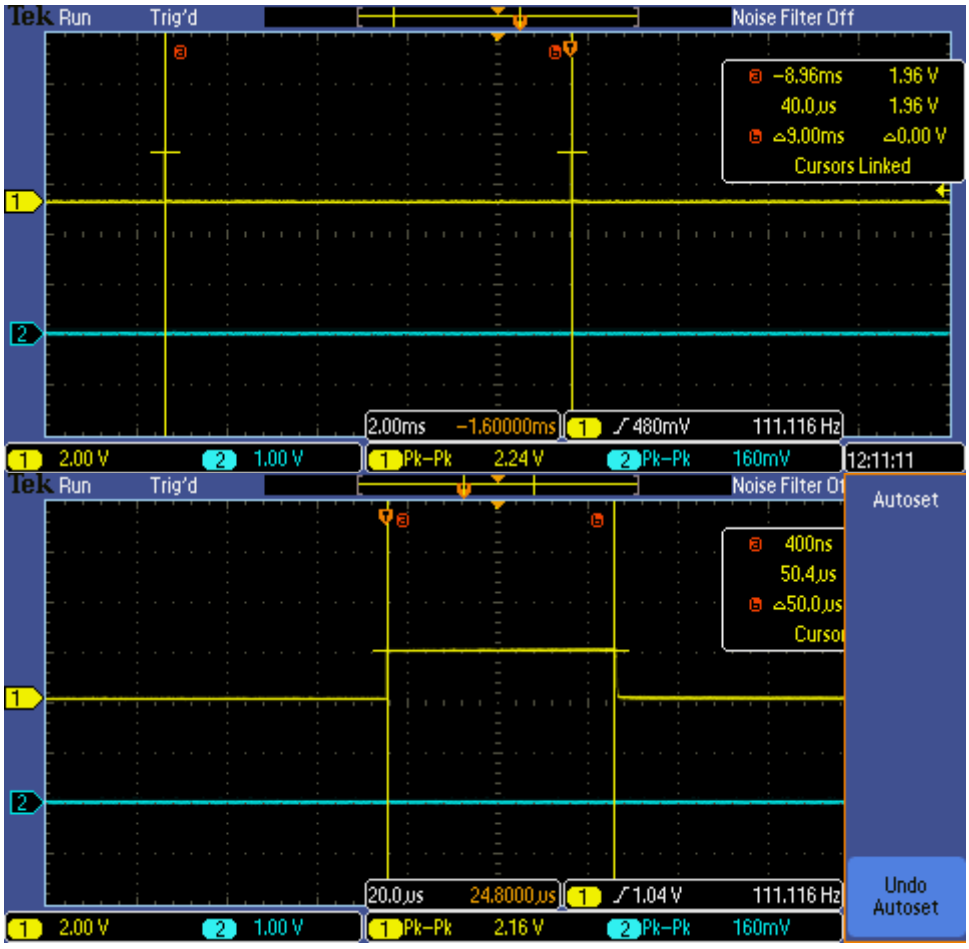

Figure 6: task 1 signal frequency and width.

For the task 2 there are 2 captures in the next figures that shows two different states for the input button.

Task 3 is the trickiest one to do. To check this, we used a DC generator to send a square signal with a frequency from 500 to 1000Hz. On the next figure there is a picture of the generator and of the signal used for the test.

Figure 7: Picture the machine used to generate a square signal.

we wait for task 9 to print and we can check if the signal is the good one. On the following figure the signal sent was a square signal with a frequency of 500Hz.

```
12:41:42.427 -> Task 2 switch's state :0, Task 3 Frequency :500.00, Task 5 analog input average :3906.75
```

Figure 8: Capture of the task 9 print from ESP Monitor serie

For task 4 and 5, we check the input the same way as for task 3: we check the signal on the task 9 print. The average digital input is 3906.75. The tested signal was 3.05V high, to pass from a digital value to an analogue one we do: $2^{12} \times \frac{3.05}{3,3} \approx 3786$ which is a closed value from the one we measured.

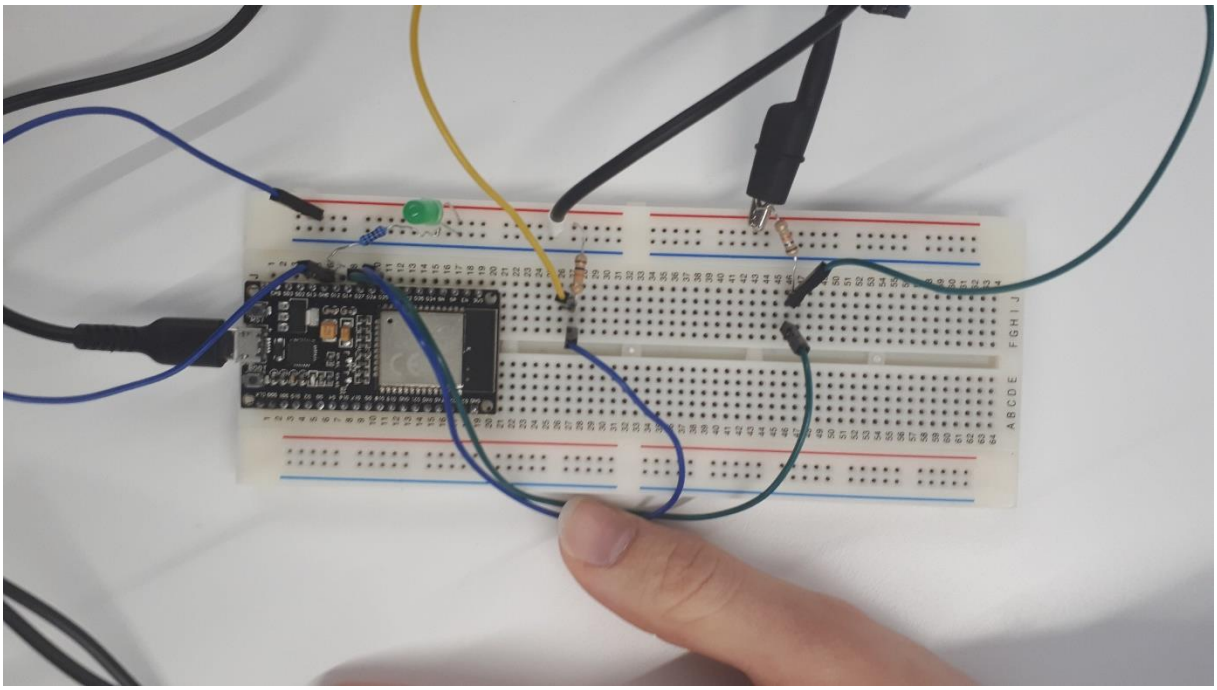

Figure 9: Montage used to test task 3,4 and 5.

For task 7 and 8 we reused the generator used for task 4 and 5. And we add an LED to see if the LED in turning on or off regarding the analogue input. On the figure 2 we can see the LED on.

For task 9 there it is simply a print every 5 seconds, this print is used to check if other tasks are working correctly like in figure 6 or in the next figure.

```
Task 2 switch's state :1, Task 3 Frequency :1000000.00, Task 5 analog input average :2286.75
```

Figure 10: Capture of task 9 with a different input analogue input and with task 2 switch on.

In conclusion, all the tasks are working properly. Furthermore, we can see that the tasks have the good timing because task 9 is printing every 5 seconds (there is not moment where this task is skipped) and on figure 6 the period of task one (the most called) is respected.

4. Git

My git repository for this class is *https://github.com/SylvainJnn/EmbeddedSoftware* . There is going to be only folder by assignments. On the figure below there is a part of the commit history.

```
PS D:\sylva\Documents\ecole - cours et autres\Cours\5eme année\Embedded Software\Assignements_git> git log
commit 82d5279ab16fe06c4a01e48f2fa9fd4a2f81762c (HEAD -> main, origin/main)
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Thu Mar 10 19:04:53 2022 +0000

    presentation version

commit b5209e53c51f68da557883d685f34f26cedffe12
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Tue Mar 8 18:03:28 2022 +0000

    add comments

commit 9daafacd6beddcc159455e145eb6b29882872075
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Tue Mar 8 13:29:05 2022 +0000

    add check_frequency function

commit 9c54433ffb7b5c528d5153a5fb15e64d14390d0d
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Mon Mar 7 13:16:22 2022 +0000

    working version

commit c2335e011ffd9d043885f62afa291012fff03125
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Fri Mar 4 19:19:40 2022 +0000

    version with only one Ticker

commit ed9c3d43f8169e24c65abf8bb6898544656e7112
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Tue Mar 1 18:37:45 2022 +0000

    cannot use multiple tickers, only 1 ticker obect -> create 'my_function' to do it with a counter

commit 9920db6f096b6cfb5bc75946808d7583bbdc3c53
Author: Sylvain Jannin <sylvainjannin@gmail.com>
Date:   Thu Feb 24 16:38:36 2022 +0000

    add all tasks, add coments, add init for all tasks
```

Figure 11 : Capture of git log repository.