# lab 6
# Application

M. Briday

October 15, 2021

## 1 Objective

The objective of this lab is to make a complete application, involving a few drivers (servo, encoder), as well as the application to manage the whole.

## 2 Servo Motor Control

### 2.1 Principle

A servomotor is a motor that integrates control electronics for position control. Thus, a position command is sent to the motor (figure 1) so that the motor axis rotates by a given angle. It is widely used in model making, with a range of use between 90°and 180°. We'll consider a range of use here of 90 degrees.

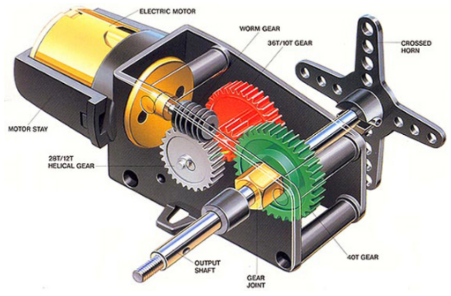
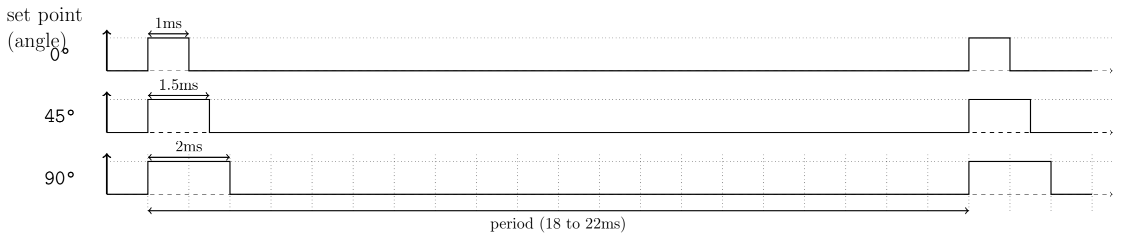


Figure 1: Servo motor

The set point sent to the actuator is a PWM signal whose high state time varies between 1 and 2ms. This instruction is sent periodically every 18 to 22ms. The diagram below shows the signal to be generated for 3 different setpoint values:



## 2.2 Driver design

We choose here a period of 20ms. The resolution of the timer will be 100$\mu$s.

2 functions should be defined to use a servo motor:

```
/* init servo motor at 45 degrees. */
void servoInit();

/* setpoint in .1% steps:
 * 0    => 0 %  =>   0 degree
 * 500  => 50%  =>  90 degrees
 * 1000 => 100% => 180 degrees
 *
 * saturations:
 * - below 0    => same as 0%
 * - above 1000 => same as 100 %
 */
void servoSet(int setpoint);
```

Instead of a configuration of the PWM directly, we develop in an incremental way, with first the configuration (and validation) of the timer, and then the PWM based on this timer.

The servo and the Led `D3` (PB0) are associated to the same pin.

**Question 1** *What is the timer that is associated to the pin of the servo? What is the alternate configuration that will be used?*

To check the timer configuration, we first program the timer to generate an interrupt each 20ms, and toggle the pin. As we do not use the PWM for now, the pin configuration should remain as OUTPUT.

**Question 2** *configure the timer. Check the frequency using the logic analyzer.*

**Question 3** *update the configuration of the timer and write the initialization function `servoInit()`. Check the signal using the logic analyzer.*

Note: The servo is powered externally. You should use an external power, set it to around 7V, and *after* plug it to the board.

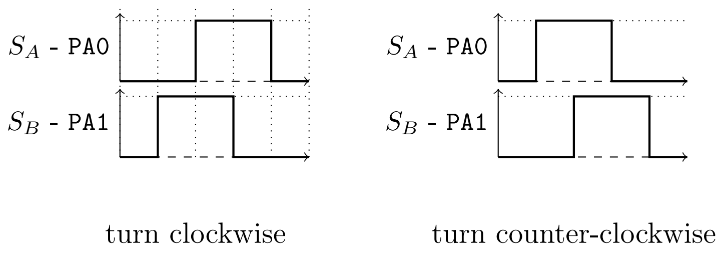


Figure 2: Pulses on $S_A$ and $S_B$ depending on the direction of rotation of the encoder.

Now, the setpoint is modified using the `servoSet()` function, as described in the comment of the function.

**Question 4** *Write the function, and write a little application that use the potentiometer to drive the servo. Check your signal using the logic analyzer and check that saturations are well implemented.*

## 3 Encoder

The encoder uses 2 signals $S_A$ and $S_B$, as defined in Fig.2. As two pulses are emitted, the software part should involve an interrupt. The driver works as follow:

- a global variable stores the current value of the encoder.
- an interrupt as associated to one signal (let's say a rising edge of $S_A$ for instance)
- in the interrupt handler, the direction is deduced from the state of the other signal
- saturations are provided on the global variable to constrain the value in $[0, 45]$.

The driver API[1] is:

```
/* init encoder  hardware. Set value to 0.*/
void encoderInit();

/* return the encoder value, in the range [0,45] */
int encoderValue();
```

**Question 5** *program and test the driver. To test the driver, you can use the TFT display.*

[1]Application Programming Interface: set of functions provided to the application programmer to interact with the driver. The API do not show all the functions, i.e. internal functions, interrupt handler and internal variable should remain hidden

## 4 Application

The application aims to control the rotation of a camera (or whatever) plugged in the servo motor (a bigger one!).

### 4.1 Manual mode

This is the simplest mode: The value reported by the encoder ($[0, 45]$) is the setpoint angle of the servo:

**Question 6** *Program (and test) the manual mode.*

### 4.2 Scan mode

In that mode, the encoder is not used any more. The servo setpoint is updated periodically each 50ms from 0% to 100% (*i.e.* 45°), with a step of 1%, and then back to 0%. As a result, it will require 5s to get from 0 to 100%, and 5 other seconds to get back.

**Question 7** *Program (and test) the scan mode. Explain how the application is tested.*

### 4.3 Both modes

We now want to switch from one mode to the other, when we press button D6 (PB1)[2].

A special attention will be paid to the transition between the 2 modes:

- from manual to scan mode, the scan mode should start from the current manual value;
- from scan to manual mode, the scan mode should not end immediately but get scan until it reaches the manual value.

**Question 8** *Program (and test) the application. Explain how the application is tested.*

[2]Of course, an FSM should be associated to the button