Robotic Workshop

TP

2 h - v1.1

IUT de Cachan, 9 Avenue de la Division Leclerc, 94234 Cachan, FRANCE

Testing sensors

Contents

1 Loading the project

This tutorial presents *testing_robots*, a simple project to test each robot component one by one. The main goal of this tutorial is for you to compile an mbed project and to understand how to use each sensor. For some sensors, you'll have to develop the function accessing the data. For others, everything is already done and the program uses the provided library.

To do 1: Import the project

1. Using **Firefox**, go on the mbed website : `https://www.mbed.com/en/` and log in or create an account.
2. Visit `https://os.mbed.com/users/gvaquette/code/workshop_tutorial/`
3. Add the project to your compiler : 

Import it into your mbed compiler before continuing. Here the project tree :

- `/`
  - `lib_workshop_2019` .... Provided library to run the robot
    - `CMPS03` .... Compass library
    - `CNY70` .... Library for reflective sensors
    - `PID` .... Library for PID
    - `Pixy` .... Library for Pixy camera
    - `VMA306` .... Library for ultrasonic sensors
  - `includes`
    - `console_output.h`
    - `pin_connexions.h` .... Declare and connect signals
    - `test_cny.h`
    - `test_compass.h`
    - `test_motor.h`
    - `test_us.h` .... for ultrasonic sensors
  - `src`
    - `test_cny` .... function files to use cny70 sensors
    - `test_compass` .... function files to use compass sensor
    - `test_motor` .... function files to run motors
    - `test_us` .... function files to use ultrasonic sensors
    - `console_output.cpp` .... function to print messages to pc.
  - `main.cpp`

The main file run an infinite loop asking the user which test does he want to run and calling the associated function.

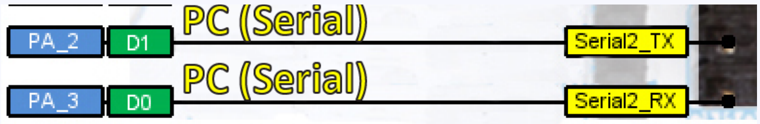
2 First connection

The project you've been given is incomplete. In order for it to compile, you'll have to declare and attach a serial connexion to your pc.

The *mbed* library provided by `"include "mbed.h"` defines the type `Serial` to declare a Serial connexion.

To do 2: Configure and connect the robot

First, we need to declare and connect the serial port to communicate with the computer. Let's check the pin map to get the pin numbers we need to connect :



1. Open the file `includes/pin_connexions.h`
2. Insert the following instruction **on line 25** in order to declare and connect the Serial port to Pins `PA_2` and `PA_3`.

```
Serial      pc      (PA_2, PA_3, 115200);
```

3. Click on the *compile* button : 

4. Save the created executable on your computer.
5. Connect the alimentation cables (red and black wires) to the robot and switch it on.
6. Connect the USB cable to the computer and the NUCLEO card.
   - The computer should detect the card and display it as a storage device.
7. Copy the executable to the card (drag and drop)
8. Open a serial communication software on your computer
   (a) Open Teraterm software (You'll find it on your desktop)
   (b) Choose Serial Connexion
   (c) Change the port to STMicroelectronics Virtual Port and hit Ok
   (d) In Setup -> Serial Port, change the baud rate to 115 200.
9. Hit the RESET button (black) on the card

3 CNY 70

3.1 The sensor

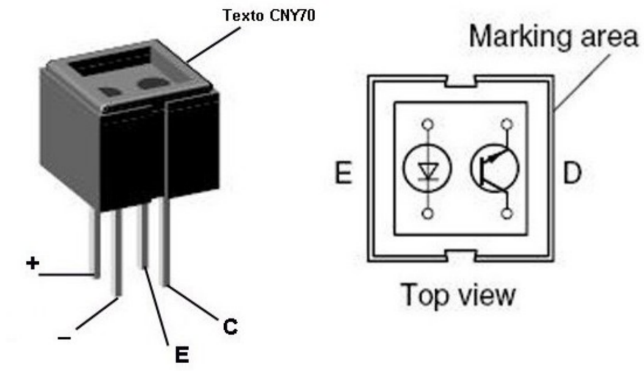


Figure 1: CNY 70

The CNY70 is a reflective optical sensor. The emitter and the detector both use resistors to be polarized. There is no clear formula for the detector as coupling factor varies with distance to the ground. Hence, we use trimmers to polarize the detector. Efficiency is a bit better with a polarization in the collector of the detector than in the emitter.

The CNY70 sensor make its detector current going up when the detector receive a light signal. To get a measure of this current, we use the voltage on the transistor's collector as shown on Figure 2.

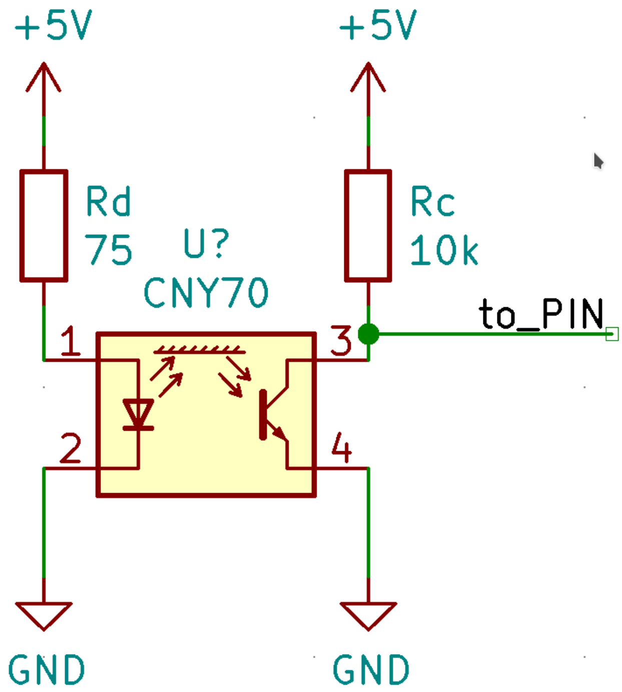


Figure 2: CNY 70 on the robot

Remarque

When the detector is on a dark surface, the voltage *to_PIN* is 5 V.
When the detector is on a white surface, the voltage *to_PIN* is low t(around 0 V).

3.2 Testing the CNY70 on the robot

First, we need to identify the PIN linked to each sensor. The three sensors are connected on pins **PA_7**, **PC_2** and **PC_3**. Since we need to measure the voltage on each pin, they need to be configured as Analog inputs.

```
case CNY70 :
    do
    {
        pc.printf("\r CNY_1");
        ft_print_cny_analog_voltage(cny_1, pc);
        pc.printf("\t CNY_2");
        ft_print_cny_analog_voltage(cny_2, pc);
        pc.printf("\t CNY_3");
        ft_print_cny_analog_voltage(cny_3, pc);
    } while (!pc.readable());
```

To do 3: Get the CNY70 voltage

1. Edit the `includes/pin_connexions.h` file to **declare the sensors:**
   - `cny_1` has already been declared : `DigitalIn cny_1 (PA_7);`
   - Declare and connect `cny_2` to **PC_2**
   - Declare and connect `cny_3` to **PC_3**
2. Edit the `main.cpp` file to match the above code.
3. Implement the function `ft_print_cny_analog_voltage(AnalogIn &analof_input, Serial pc)` in file `src/test_cny/ft_print_value_cny.cpp` :
   (a) Declare `value` and `voltage` variables (`double`)
   (b) Use the function `analog_input.read()` to get the converted analog value of the `analog_input` signal, put it in `value`.
   (c) compute `voltage = value * max_voltage;` to convert it.
   (d) Use the `pc.printf("Voltage value : %lf ", voltage);` function to display the voltage value.
4. Compile, download and transfer the executable file into the card.
5. Identifie which sensor is connected to which signal.
   - put a white sheet in front of each sensor and see which one react.

4 Controlling the motors

Our robot moves with two DC motors.In the simplest way, the DC motor speed is proportionnal to its input voltage U. Since we want to control the motor speed, we need to control the input voltage. An efficient and quite simple way to control this voltage is using Pulse Width Modulation (PWM).

4.1 Pulse Width Modulation

Définition

The duty cycle is defined as the fraction of one period in which a signal or system is active.

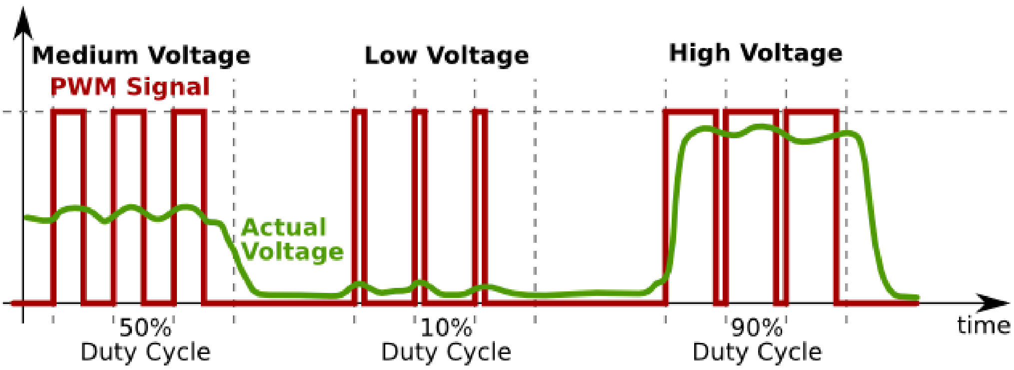


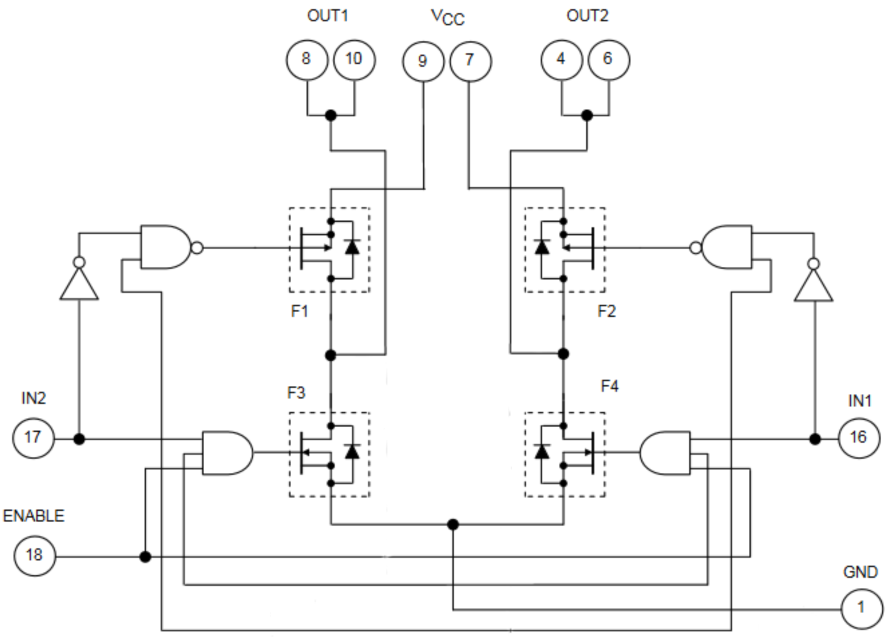
Figure 3: PWM signal

The idea is to make a high frequency signal and control its average value to the voltage value we want.

The higher the duty cycle is, the higher the signal average will be, as illustrated on Figure 3. If the duty cycle is 0,5 (50 %), the motor speed is at 50 % of its maximal value.

4.2 Controling the robot's motors : Full bridge

To control the robot's motors we use a full bridge, illustrated on Figure 4. As indicated in the table on Figure 4, `IN_1` and `IN_2` control the direction. We control the speed by connecting the PWN to the `ENABLE` pin.



(a) Schematic

Motor Drive Conditions (H: High-level input; L: Low-Level Input)

	IN1	IN2	ENABLE	Remarks
Stop	H	L	L	Turns the power supply OFF. ENABLE must be set Low when V_{DD} is rising or falling.
	L	H	L	
	H	H	L	
Forward (CW)	H	L	H	No input signal is needed that turns off the upper- and lower-side drive devices when switching the rotational direction.
Reverse (CCW)	L	H	H	
Brake	L	L	L or H	GND side MOSFET ON
	H	H	H	V_{CC} side MOSFET ON

(b) Control

Figure 4: Full bridge

4.3 Testing the motors

For testing if the PWM controlling the motors is fully functionnal, we use the `ft_test_motor` function. This function ask the user to define the wanted duty cycle and call the `ft_run_motor` function to run the PWM. You will implement this function to run the motors :

```
void ft_run_motor(  e_direction direction, double duty_cycle,
                    PwmOut pwm_mot, DigitalOut  dirA, DigitalOut  dirB)
{
    if (direction == FORWARD)
    {
        // To be completed
    }
    else /* (direction == BACKWARD) */
    {
        // To be completed
    }

    /* TO DO : apply duty cycle to pwm_mot */
    // To be completed
}
```

To do 4: Testing the motors

1. On the pinmap of the nucleo card (Figure 6 and Figure 7), look for *L. Mot Dir1*, *L. Mot Dir2*, *R. Mot Dir1* and *R. Mot Dir2* and write the associated PIN.
2. In the mbed compiler, edit the file `includes/pin_connexions.h`
   (a) Declare and connect the `DigitalOut` signals `DIR_1L`, `DIR_2L`, `DIR_1R`, `DIR_2R` to control the direction of left and right motor.
      - `DigitalOut DIR_1L (PC_9);`

   (b) Declare and connect the `PwmOut` signals `Pwm_ML and Pwm_MR` to control the PWM of each motor.
      - `PwmOut Pwm_ML (PA_9);`
3. In the mbed compiler, edit the `Src/test_motor/ft_run_motor.cpp` file so that
   (a) Set the `dirA` and `dirB` value depending on the wanted direction,
   (b) Set the PWM signal to the `duty_cycle` given in argument.
4. Edit `main.cpp` to call `ft_test_motor` function:

```
case LEFT_MOTOR :
    ft_test_motor(Pwm_ML, DIR_1L, DIR_2L, pc);

case RIGHT_MOTOR:
    ft_test_motor(Pwm_MR, DIR_1R, DIR_2R, pc);
```

5 Testing other sensors

To do 5: Testing other sensors

Use the interface to test if every sensor is working.

5.1 VMA 306 : Ultrasonic sensor

To use the VMA306 sensor, we set the *Trig* pin to **high** for at least 10 µs, then the sensor send a burst of 8 x 40KHz pulses and set the *echo* pin to **high**. When the burst comes back to the sensor (caused by the echo), the sensor set its *echo* pin to **low** (illustrated on Figure 5).

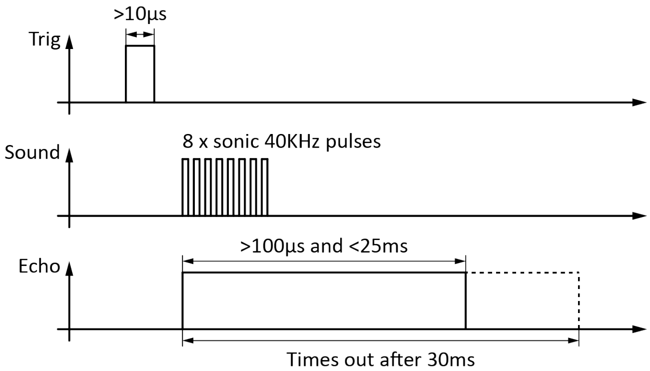


Figure 5: VMA306 sensor

Thus, the time during which the *Echo* pin is **high** represent the time the sound traveled. Let t_{up} be the time of *Echo* being **high** and $v_s = 340\,\text{m/s}$ the sound speed. The measured distance d can be expressed as

$$d = v_s \times t_{\text{up}}$$

To avoid false detection each VMA306 send his pulse every 150ms and with a 50ms delay after previous one. There are 3 VMA306 on the robot: one on the left, one on the front, one on the right.

In the provided library, you can access to VMA306 data.

6 Pin map of the robots

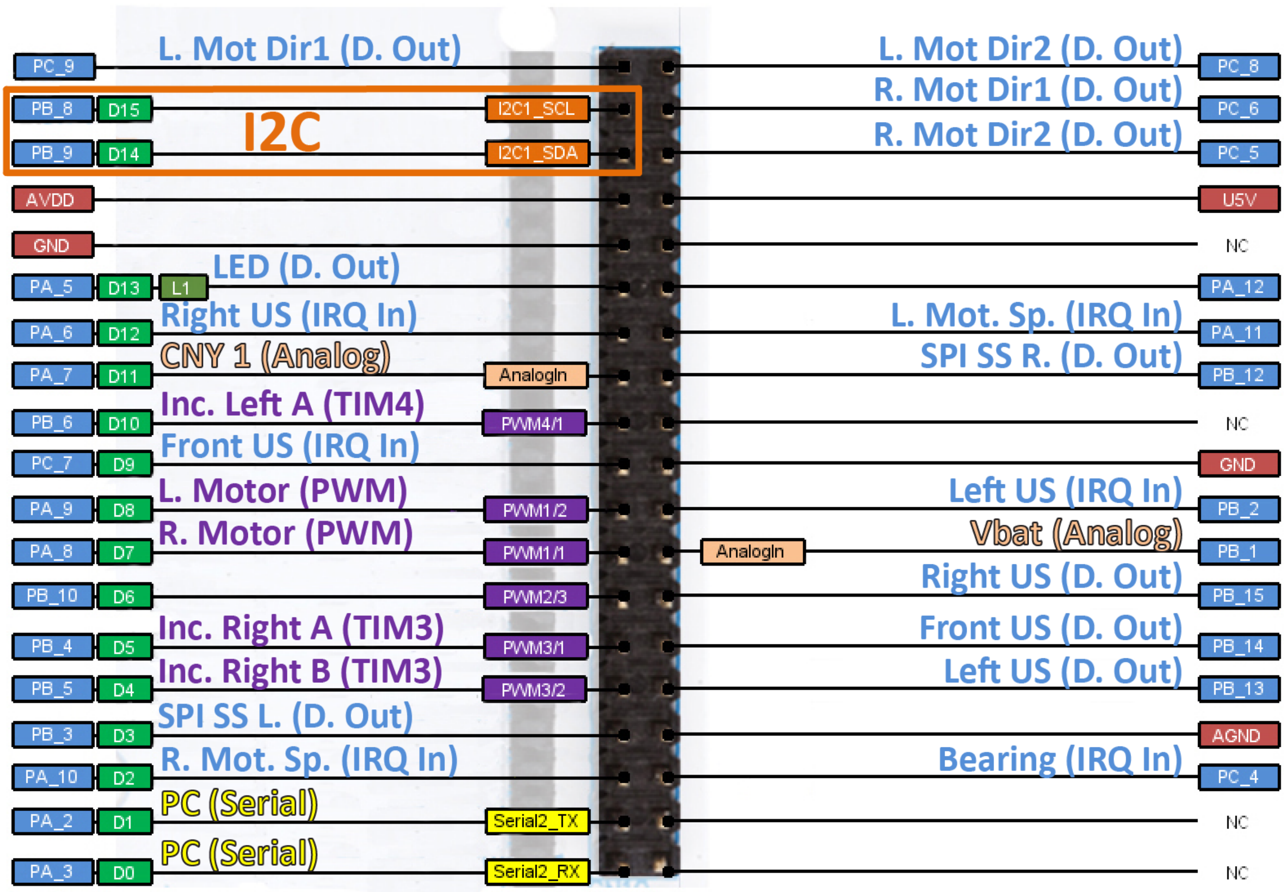


Figure 6: PIN map of the right connector

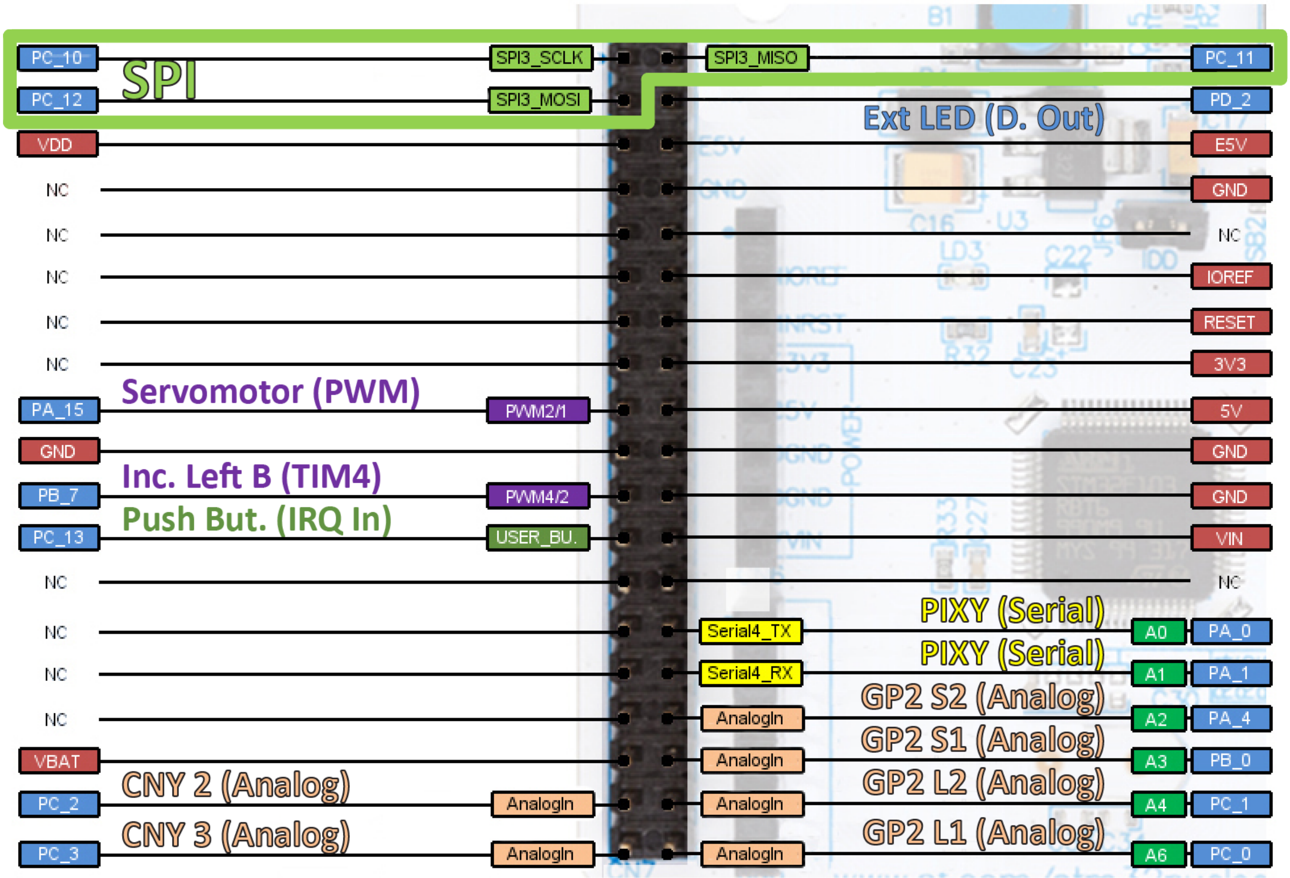


Figure 7: PIN map of the left connector