Universidade do Minho
Escola de Engenharia

STM32 - Control of a submersible

Integrated Practices Laboratory 2

Group 1:
André Campos 83200
Daniel Fernandes 81700
Fábio Cunha 80200
João Sousa 82273
Samuel Pereira 81408
Pedro Sousa 82041

Guidance teachers:
Paulo Carvalhal
Luís Gonçalves

Index

1. Functional Specifications

1.1 Functional Description

The product developed consists of a submersible for underwater research. This product is capable of acquiring physical data underwater, at different depths and positions. This product is divided into two development stages:

- Firstly, the implementation of the basic structure of its depth control system. A pressure sensor that converts the force exerted by the water into a voltage directly proportional to the depth and a sonar that allows it to have a better perception of its surroundings, thus avoiding

collisions/damage. These entities correspond to the controlled variables and will be one of the inputs to the control subsystems.
In order to create a variable weight and make submersion/immersion possible, a linear actuator is used to move the water in and out of the reservoir (syringe).

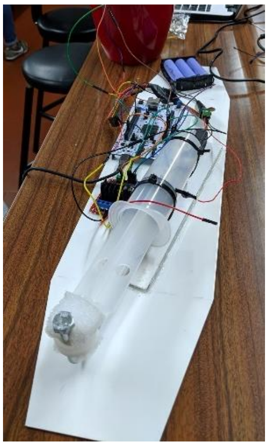
-The second stage, in a future product development, involves the control and directional movement/driving of the submarine and optical visualisation underwater. Thrusters and a motor are used for movement in the horizontal plane, and a camera transmits the image to the application.

As the wireless communication in the aquatic environment that currently exists are inefficient and precarious methods, communication is done by (long) wire between the object and the application. A high-performance, long-lasting power supply is incorporated separately (developed last semester)

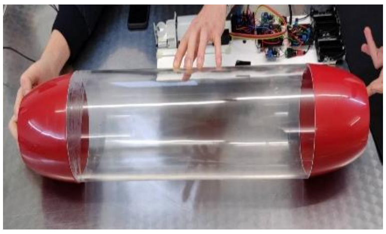
The final product represents an offensive methodology, where the research and science market for research purposes are the main targets, being a versatile product with a long lifespan, differentiating it from the competition. Even influenced by the project's prerequisites, this strong conviction remains in the group's thinking.

1.2 Submersible user manual

1. Submersible assembly:
   -Internal circuit inside the housing;
   -Connect cables to the outside (communication, syringe) to their respective outlets;
   -Proceed to isolation;

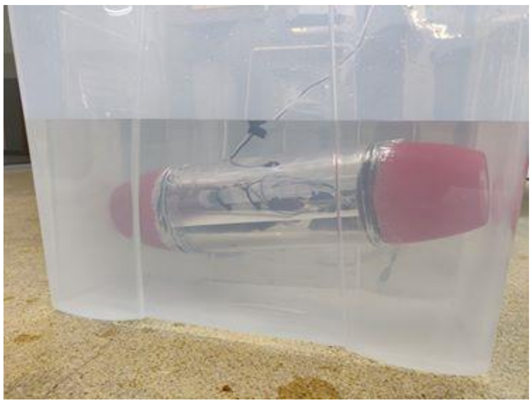


+



2. Monitoring programme link:
   -Connection of the USB extenders for the microcontroller and camera;
   -Run programme;

3. Start the Submersible:
   -Place submerged in water (max. depth 20m);
   -Magnet -> ON/OFF via a magnetic switch;
   -Set the desired depth on the interface;
   -Monitor graph values;



2. Technical Description

Here is a model summarising how the submersible works:

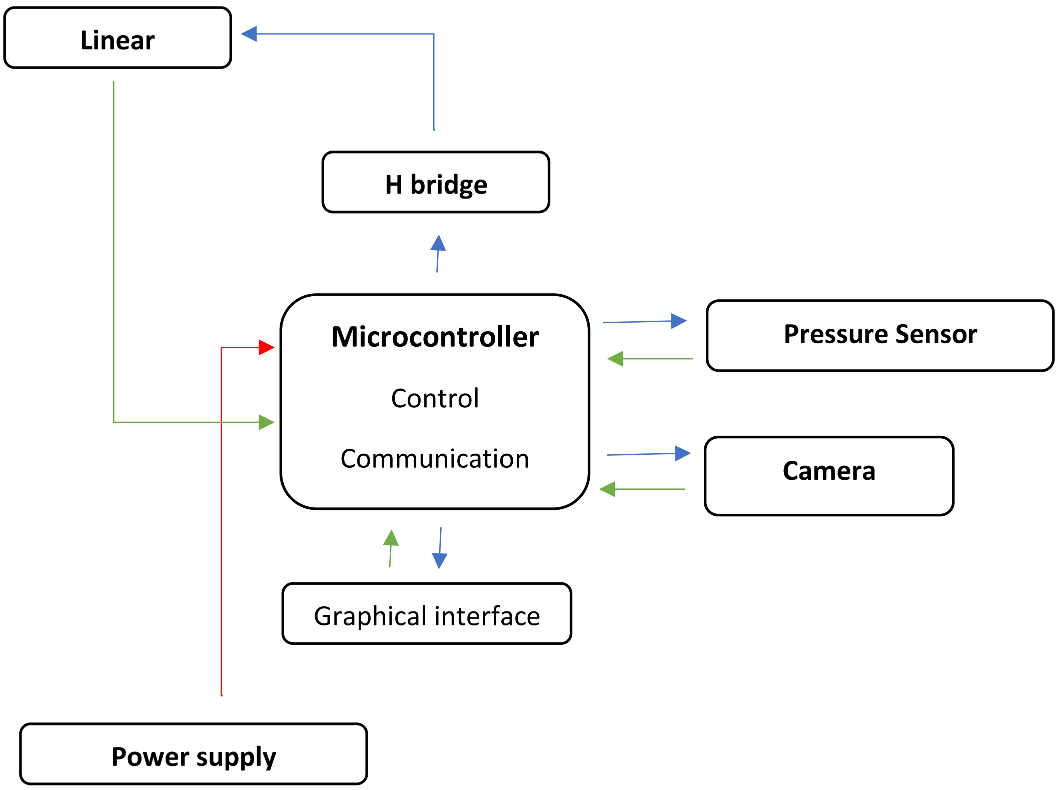


Figure 1 - Modules/Subsystems

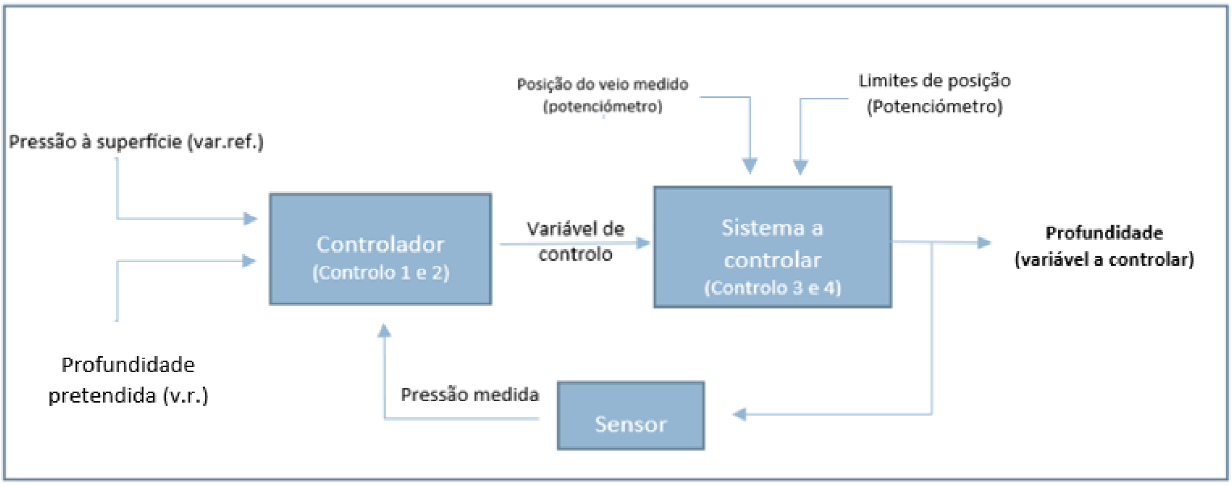


Diagram 1 - Control Diagram

- Control 1 and 2: relating to the calculation of pressures and the differentiation of depths/pressures;

-Control 3 and 4: Protection, comparison and actuation;

2.1 Control/Actuation

Control is carried out using the PID control technique taught in the Control course. The proportional integral derivative (PID) controller combines 3 different types of action:

- Proportional - adjusts the control variable in proportion to the error.
- Integral - adjusts the control variable based on the time the error occurs, i.e. proportional to the magnitude and duration of the error.
- Derivative - adjusts the control variable based on the variation in error. The combination of these types of control forms the controller known in the industry as PID.

The ideal PID controller in parallel form, defining u(t) as the output signal, can be defined by:

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

Where u(t) is the control variable, e(t) is the error variable, Kp is the proportional gain, Ki the integral gain and Kd the derivative gain. The error variable is equal to the reference variable minus the controlled variable, e(t) = yr(t) - y(t).

To digitally approximate the previous expression in time, with a sampling period of h, the following expression can be used in discrete time:

$$u(k) = K_{p\_h} e(k) + K_{i\_h} \sum_{i=1}^{k} [e(i) + e((i-1))] + K_{d\_h}[e(k) - e((k-1))]$$

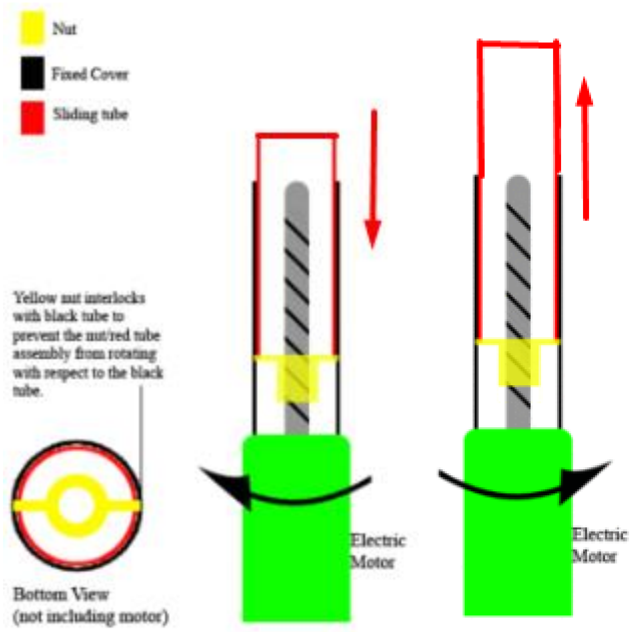
As for its technical specifications:

Control algorithm	**PID**
Entry	
Maximum voltage	3.3 V
Maximum current	25mA
Sampling frequency	X Hz
Output	
Tension	0 to 9 V
PWM frequency	1 Khz

2.2 Motor and its Actuator

The motor used is a linear actuator (an Actuonix) measuring 15 cm and extendable to 30 cm, with a position feedback potentiometer.

The linear actuator creates movement in a straight line by utilising the circular movement of the electric motor with a linear converter.



The actuator is attached in parallel to a 100ml syringe (equivalent to approximately 100g when filled with water).

With a force of 15N needed to introduce or expel water, the motor is quite capable of fulfilling its role.

When its poles are positively charged, the shaft extends, just like the syringe to which it is attached, and the water is sucked into the syringe. (Weight gain)

When fed negatively, the shaft and syringe collect and the water is expelled.

The motor also has an internal potentiometer capable of transducing the position of its shaft to a voltage of +5V to 0V. This potentiometer is used in the position control circuit and the protection circuit.

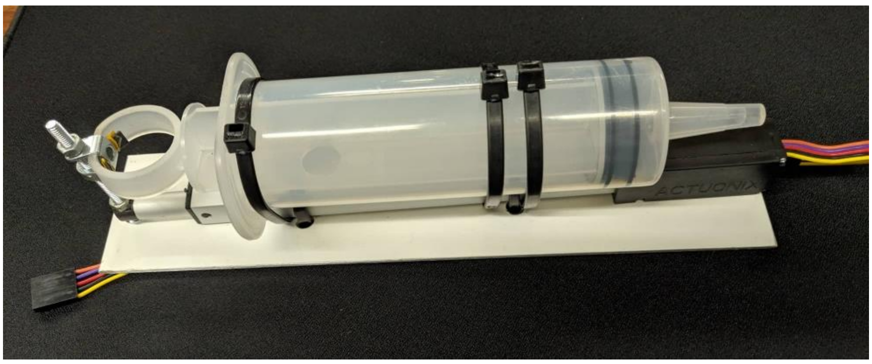


Figure 3 - Actuator and 100 ml syringe attached

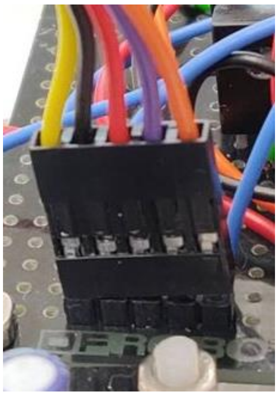
1 - Orange - Rail negative position reference feedback potentiometer;

2 - Purple - Rail position feedback potentiometer;

3 - Red - V+ motor (6V or 12V);

4 - Black - Motor V- (*Ground*);

5 - Yellow - Positive rail position reference feedback potentiometer;



L16 Specifications

Gearing Option	150:1
Peak Power Point	175N @4mm/s
Peak Efficiency Point	75N @7mm/s
Max Speed (no load)	8mm/s
Max Force (lifted)	200N
Back Drive Force	102N
Stroke Option	**140mm**
Mass	84g
Repeatability (-P & LAC)	0.5mm
Max Side Load (extended)	20N
Closed Length (hole to hole)	218mm
Feedback Potentiometer	16kΩ±50%
Feedback Linearity	Less than 2.00%
Input Voltage	0-15 VDC. Rated at 12VDC.
Stall Current	650mA @ 12V
Operating Temperature	-10°C to +50°C
Audible Noise	60dB @ 45cm
Ingress Protection	IP-54
Mechanical Backlash	0.25mm
Limit Switches	Max. Current Leakage: 8uA
Maximum Static Force	250N
Maximum Duty Cycle	20%

Table 1 - Linear actuator specifications

2.3 Submersible housing

The final capsule designed for the submersible is a transparent acrylic tube with a diameter of 14 cm and a length of 35 cm. At the ends, there are 2 caps that increase the length of the module to 63 cm.

Two openings in the acrylic are made for the communication wires with the external tower and the water inlet/outlet pipe. Another opening in the lid for the sensor to be in contact with water. Once fitted and covered with insulating tape, we have our waterproof capsule.

Overall, the total volume of the capsule is around 12.4 dm^3, which equates to a force of 85 N (8.5 kg) to sink it, compensating for this lack of weight with four 2 kg weights.

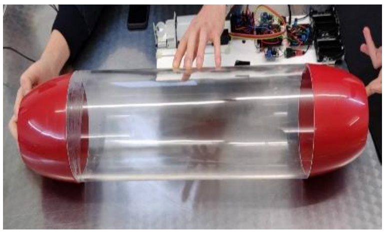


Figure 9- Capsule and lids

2.4 Pressure Sensor

The MS5837-30BA is a high-resolution pressure sensor with a range of 0 to 30 Bars (unit of pressure measurement) with I2C Interface. It has high linearity and a 24-bit ADC with self-calibration. To measure pressure, it uses an algorithm that involves measuring temperature and some associated calculations.

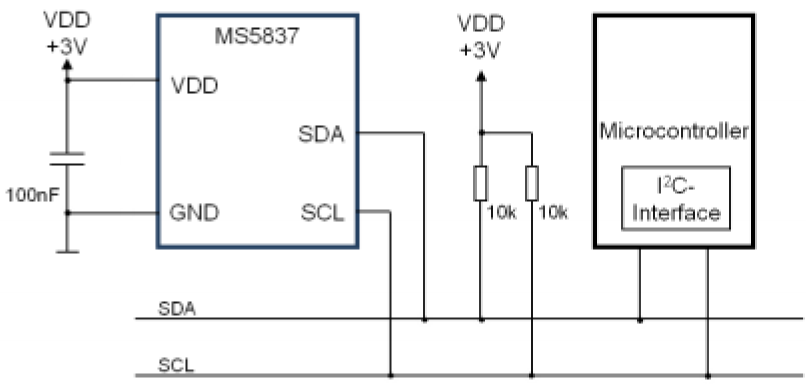


Figure 4 - I2C bus assembly wiring diagram

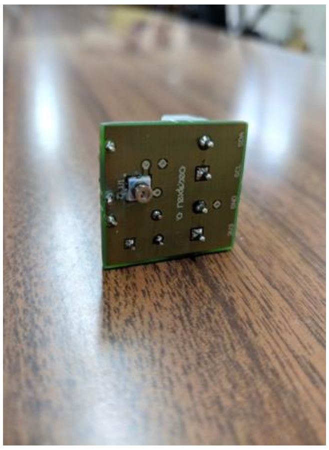
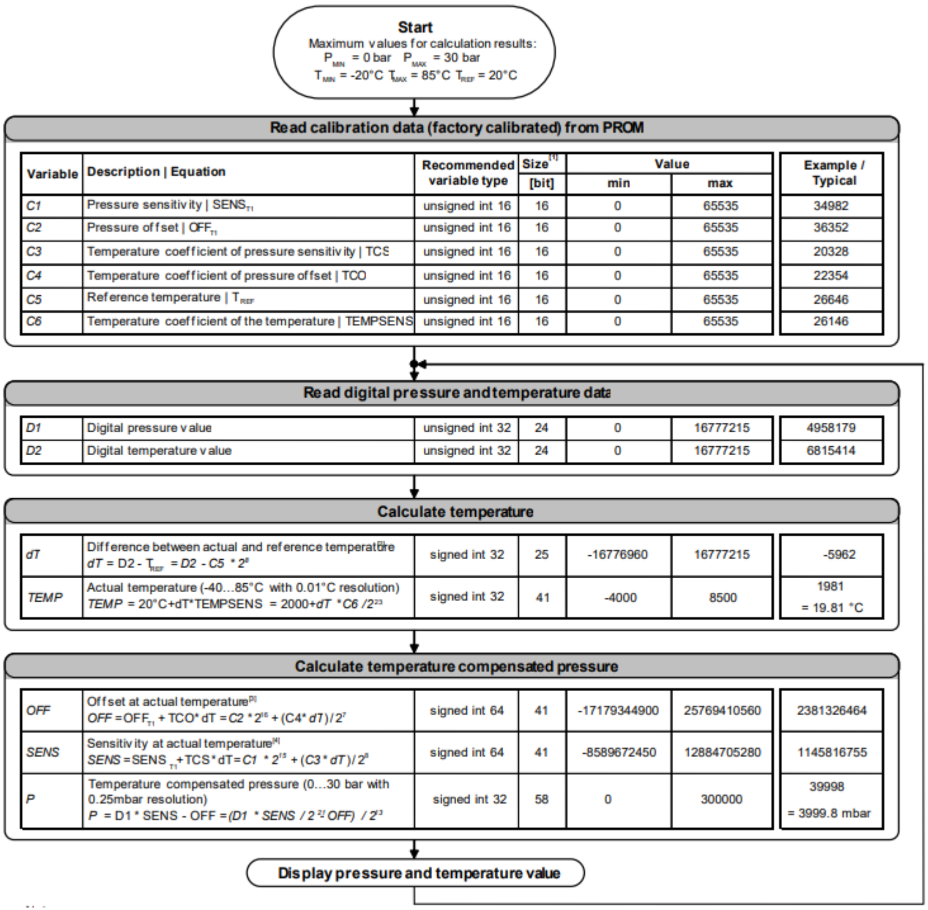


Figure 5 - Bottom of the integrated circuit

Figure 6 - Top of the integrated circuit with the necessary components for the I2C line

As soon as the sensor starts working, it sends the calculation coefficients via the I2C bus, which are then used to calculate both temperature and pressure.



Start
Maximum values for calculation results:
P_{MIN} = 0 bar P_{MAX} = 30 bar
T_{MIN} = -20°C T_{MAX} = 85°C T_{REF} = 20°C

Read calibration data (factory calibrated) from PROM

Variable	Description | Equation	Recommended variable type	Size[1] [bit]	Value min	Value max	Example / Typical
C1	Pressure sensitivity | $SENS_{T1}$	unsigned int 16	16	0	65535	34982
C2	Pressure offset | OFF_{T1}	unsigned int 16	16	0	65535	36352
C3	Temperature coefficient of pressure sensitivity | TCS	unsigned int 16	16	0	65535	20328
C4	Temperature coefficient of pressure offset | TCO	unsigned int 16	16	0	65535	22354
C5	Reference temperature | T_{REF}	unsigned int 16	16	0	65535	26646
C6	Temperature coefficient of the temperature | TEMPSENS	unsigned int 16	16	0	65535	26146

Read digital pressure and temperature data

D1	Digital pressure value	unsigned int 32	24	0	16777215	4958179
D2	Digital temperature value	unsigned int 32	24	0	16777215	6815414

Calculate temperature

dT	Difference between actual and reference temperature $dT = D2 - T_{REF} = D2 - C5 \cdot 2^8$	signed int 32	25	-16776960	16777215	-5962
TEMP	Actual temperature (-40…85°C with 0.01°C resolution) $TEMP = 20°C + dT \cdot TEMPSENS = 2000 + dT \cdot C6 / 2^{23}$	signed int 32	41	-4000	8500	1981 = 19.81 °C

Calculate temperature compensated pressure

OFF	Offset at actual temperature[3] $OFF = OFF_{T1} + TCO \cdot dT = C2 \cdot 2^{16} + (C4 \cdot dT) / 2^7$	signed int 64	41	-17179344900	25769410560	2381326464
SENS	Sensitivity at actual temperature[4] $SENS = SENS_{T1} + TCS \cdot dT = C1 \cdot 2^{15} + (C3 \cdot dT) / 2^8$	signed int 64	41	-8589672450	12884705280	1145816755
P	Temperature compensated pressure (0…30 bar with 0.25mbar resolution) $P = D1 \cdot SENS - OFF = (D1 \cdot SENS / 2^{21} - OFF) / 2^{13}$	signed int 32	58	0	300000	39998 = 3999.8 mbar

Display pressure and temperature value

Figure 7 - Pressure and temperature calculation algorithm on the sensor *datasheet*

It was on the basis of this algorithm that we calculated the variables we needed (pressure, temperature). Firstly, the sensor determines the coefficients needed for subsequent calculations and sends them via the I2C bus to the STM32 board.

2.5 Power Supply and Protection

To power the prototype's main circuit (linear actuator) we use a set of 3 lithium batteries (totalling approximately 12V, capable of powering the motor). This set of batteries is protected by a device called a BMS (Battery management system), whose function is to protect the lithium batteries by keeping them within their safe operating range (voltage and current conditions under which the device can operate without any damage).

Since all the components have to be inside the prototype, i.e. it is impossible to power the prototype from external sources, the following is used:

Figure 9 - 3.6V 2.5Ah batteries

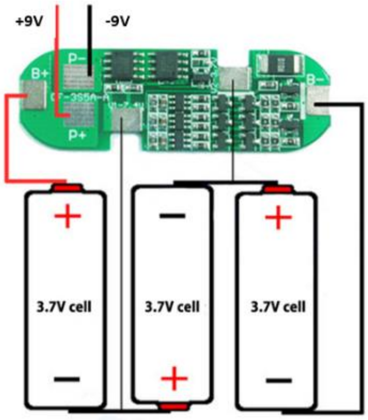


Figure 8 - Connecting the BMS to the battery pack

Figure 9 - Batteries+BMS

2.6 Application Graphical Interface

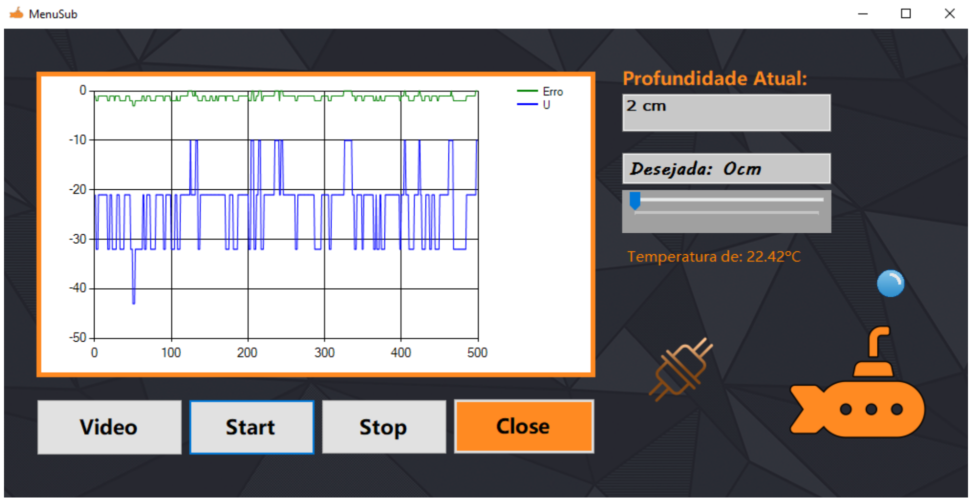


Figure 8 - Graphical interface (control window)

The interface application was developed in C# using Visual Studio software. Communication between the graphic interface and the prototype is done by wire via the serial port.

The graph shows the evolution of the control and error variable over time. It also has 2 labels:

-One to show the current depth of the submersible;

-Another that shows the value of the reference variable (desired depth) controlled by the small blue cursor.

You'll also notice that the temperature at which the water in which the submarine is sailing is being displayed (calculated by the sensor).

Buttons:

- Start - starts the programme;
- Stop - suspends any action of the programme;
- Close - ends the programme;
- Video - shows a window with video from the prototype camera and allows you to take a photo;

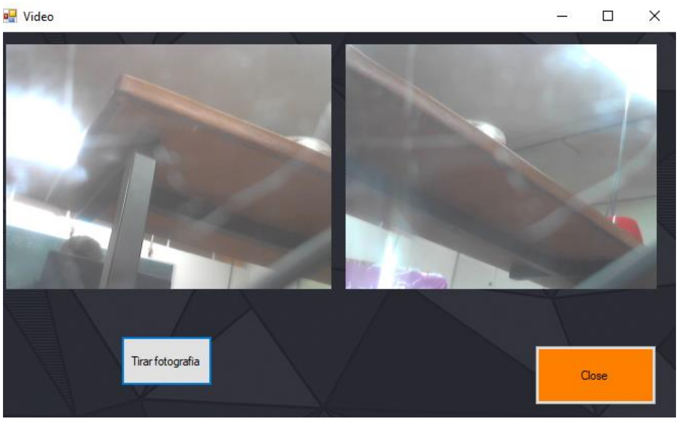


Figure 10 - Camera images in the graphical interface

- ON/OFF sensor - represents whether the sensor is in perfect condition and communicating with the STM32 board.

Figure 9 - Sensor disconnected

3. Description Microcontroller Implementation

3.1 PID control

Implementation of the PID control explained above:

```
void controlador(uint8_t valor_ref, uint8_t valor_atual){

e=valor_ref-valor_atual;
	sum_e_bkp = sum_e; //error sum backup
	sum_e = sum_e+e+e_ant;

	u_d = ( Kd_f * (e-e_ant) )+ K_f*u_d_ant;  //u_d
	e_ant = e;
	u_d_ant = u_d;
	delta_u = Kp_h*e + ( Ki_h * sum_e )+ u_d;  //delta_u
	u = u_0 + delta_u;

	if(u>U_sat_a){ //u above upper saturation
		u=U_sat_a;
		sum_e=sum_e_bkp;  //sum of errors frozen,
		                  //back to last value
		}
	else if(u<U_sat_b){ //u below lower saturation
		u=U_sat_b;
		sum_e=sum_e_bkp;   //sum of errors frozen,
		                  //back to last value
		}
	output_t(u); //unsaturated u remains unchanged ------ A DEFINIR


e_ant=e;
     }
```

3.2 Function of action

Sets the direction and duty-cycle value of the PWM acting on the motor, depending on the calculations made in the control:

```
void output_t(int atuador){

    if(atuador>0){ //u above upper saturation
      FW_RV = 1;
      value = atuador;
}
    else if(atuador<0){ //u above upper saturation
      FW_RV = 0;
      value = atuador*(-1);
}

      if(atuador>0 && !get_stop()){ //u above upper saturation
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);
        htim2.Instance->CCR1 = value;
}

      else if(atuador<0 && !get_stop()){ //u above upper saturation
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_SET);
        htim2.Instance->CCR1 = value;
}
```

3.3 Engine limitations

This limitation prevents the motor from exceeding the extremes of the syringe by comparing the motor's current value with the limit values:

```
void limitacao_f(float valor_adc){

    //printf("CP %d j \r\n",(int)valor_adc);
    if((valor_adc < VAL_MIN_LIMIT) &&!get_sentido()){
        //htim2.Instance->CCR1 = 0;
        stop = 1;
    }

    else if((valor_adc > VAL_MAX_LIMIT) && get_sentido()  ){
      //htim2.Instance->CCR1 = 0;
      stop = 1;
    }
    else{stop = 0;}

}
```

3.4 Main.c:

Configures and initialises all the peripherals, and runs the following code in a loop:

```
while (1)
{


if(test_graf){
printf("CG %d i \r\n",10 );
HAL_Delay(2);
HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_14);

}
if(messageReceived()){
    interface();
}
```

Checks the commands entered,

```
void interface(void)
{
    if(COMMAND[0] == 'C' && COMMAND[1] == 'Y')
  {
    test_graf=1;
  }

    if(COMMAND[0] == 'C' && COMMAND[1] == 'S')
  {
    test_graf=0;
  }

      if(COMMAND[0] == 'S' && COMMAND[1] == 'E')
  {
    float aux3;

    if(exec_getValue(&aux3)){
    valor_referencia = (int)aux3;
    printf("CP %d j \r\n", valor_referencia );
    }
  }

}
```

CY - Activates flag to start the readout graph

CS - Deactivates read graph flag

SE -Get() and prints the pressure value;

4. Component listings and prices

COMPONENT	PRICE
Acrylic tube	40€
Linear actuator	88€
Lithium batteries(x3)	11,95€
Regulator	3€
Glue tape	2,5€
USB - micro USB cable	3€
Weights	##€
H-bridge	9€
Pressure sensor	30€
Camera	6€
Battery charger	8€
Syringe	2€
BMS	3,75€
Others	Approximately 20€
Total	204.5

5. Safety, Reliability and Certification Study

In terms of safety, due to the very low voltage and current levels, the submersible would be unlikely to cause any physical harm to a person, assuming the equipment is used responsibly and correctly.

In terms of material (external) damage, the risk is also minimal. In terms of the internal components, in the event of a waterproofing failure, it would damage the electrical components and cause extensive and irreparable damage.

In terms of reliability, the submersible, under rated conditions and with proper use, will have a long lifespan since all the components have many hours/cycles of expected life.

In terms of certification, since all the components have been purchased within the European Union, and since they have to be regulated in accordance with all safety, hygiene and environmental protection standards in order to be marketed in the European economic area, our product is also included.