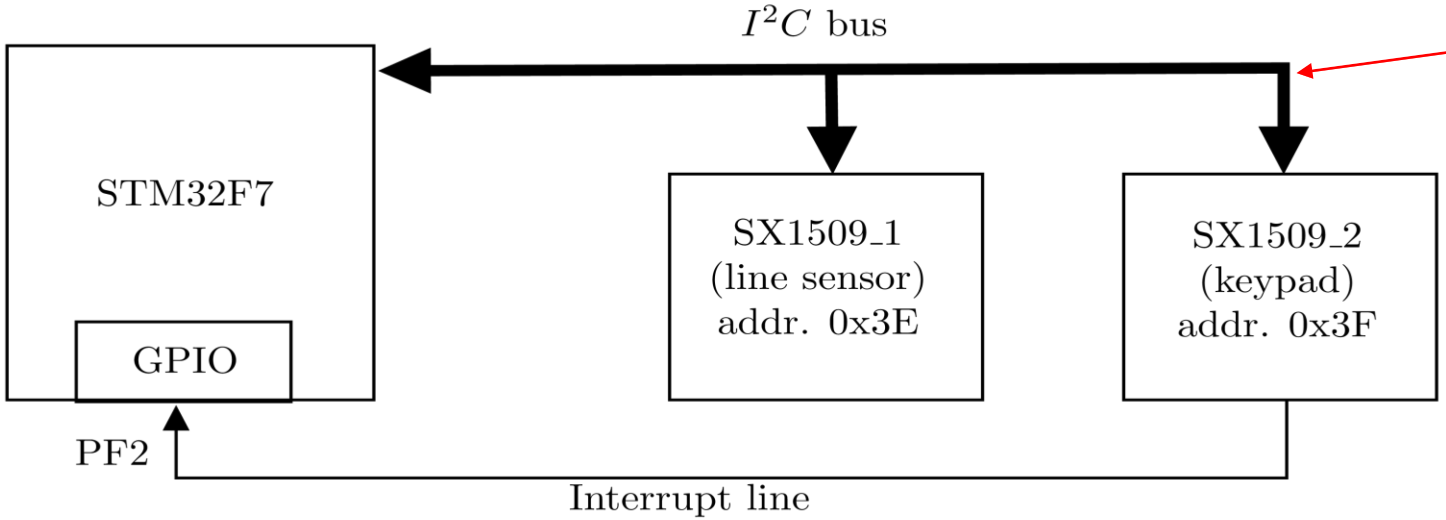
Embedded Real-Time Control

A.A. 2022/2023

Bettin Paolo - 2089152
Vitetta Emanuele - 2082149
Merolli Martina – 2072012
Guglielmin Giorgia – 2088623
Bonaventura Luca - 2090005

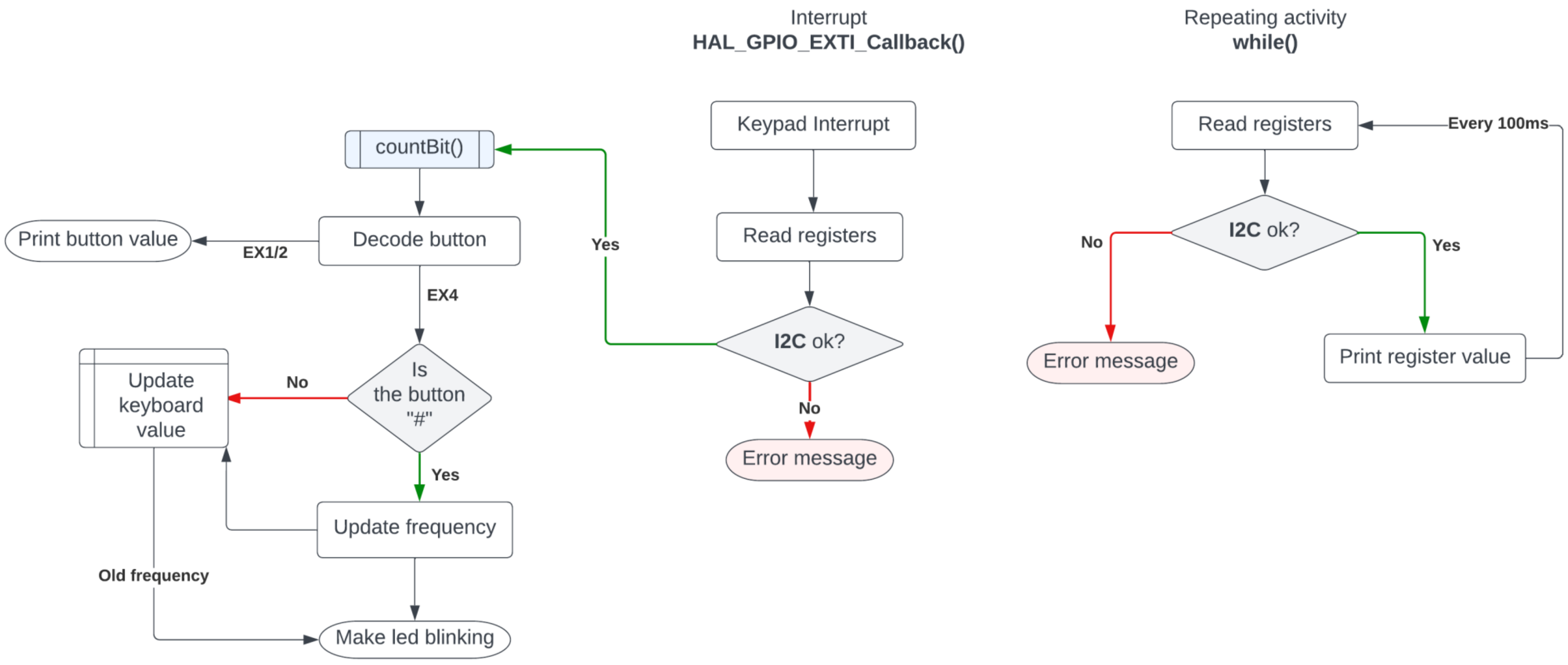
Laboratory 1 – Basics

Schematics of the I2C communication interface in the TurtleBot for the register reading



We are going to read **two** registers

Program flowchart



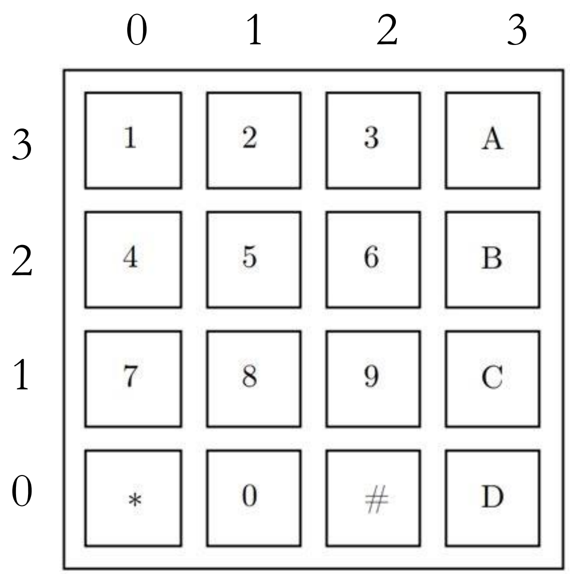
Implementation details

Char To Decimal Conversion

Dec	Hex	Char
32	20	Space
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9

X [dec] = X [char] - 48

Keypad



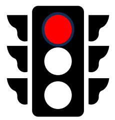
Rows and Columns in C matrix

CountBit()

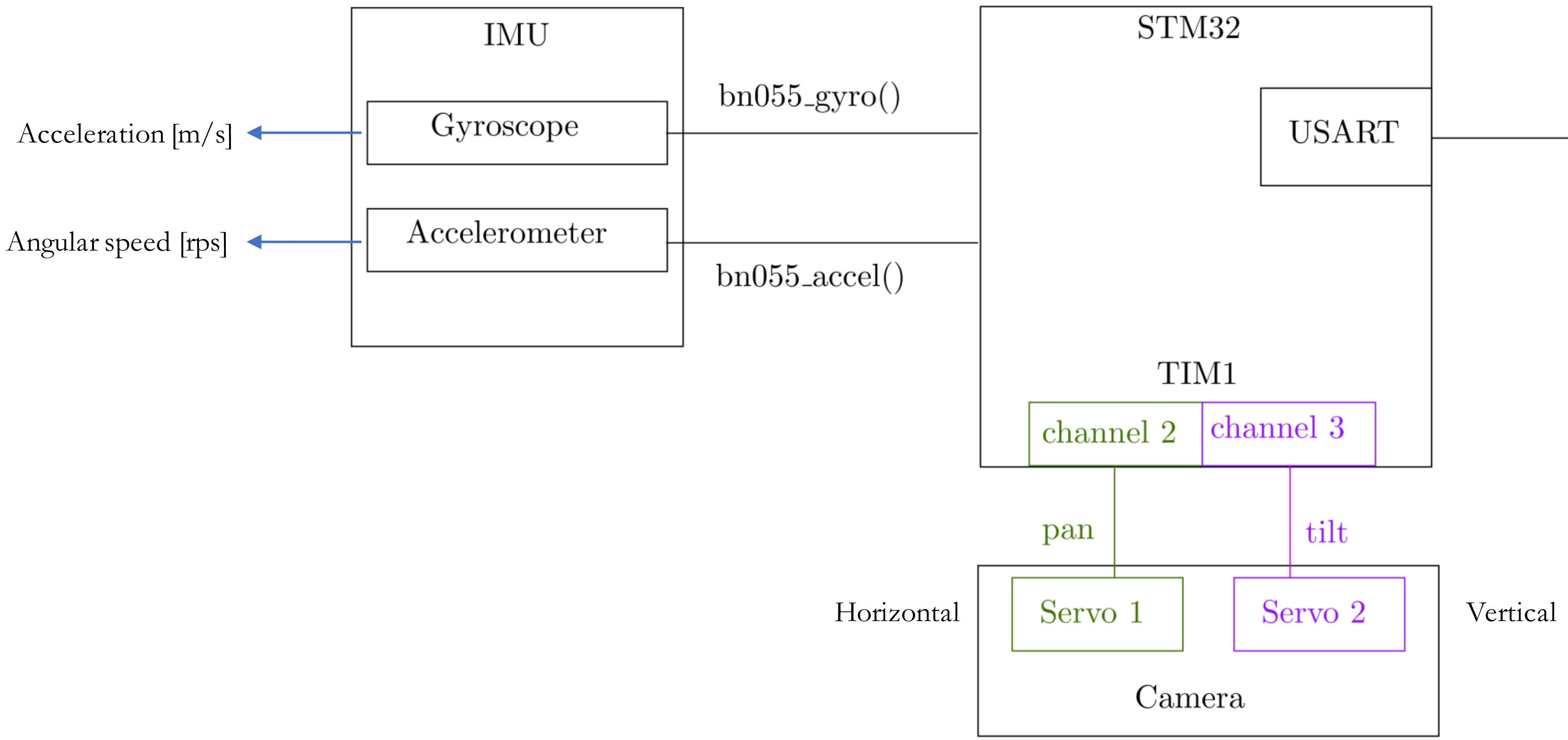
- 1101 — Original data
- 0010 — Bitwise negation
- 0001 — Decrease by one because indexes starts from 0
- 1 — Count the number of 1 in the string (AND + right shift)

Example for row/column 1 (only 4 out of 8 bits reported)

About laboratory 1 Extra...



Laboratory 2 – Camera stabilizer

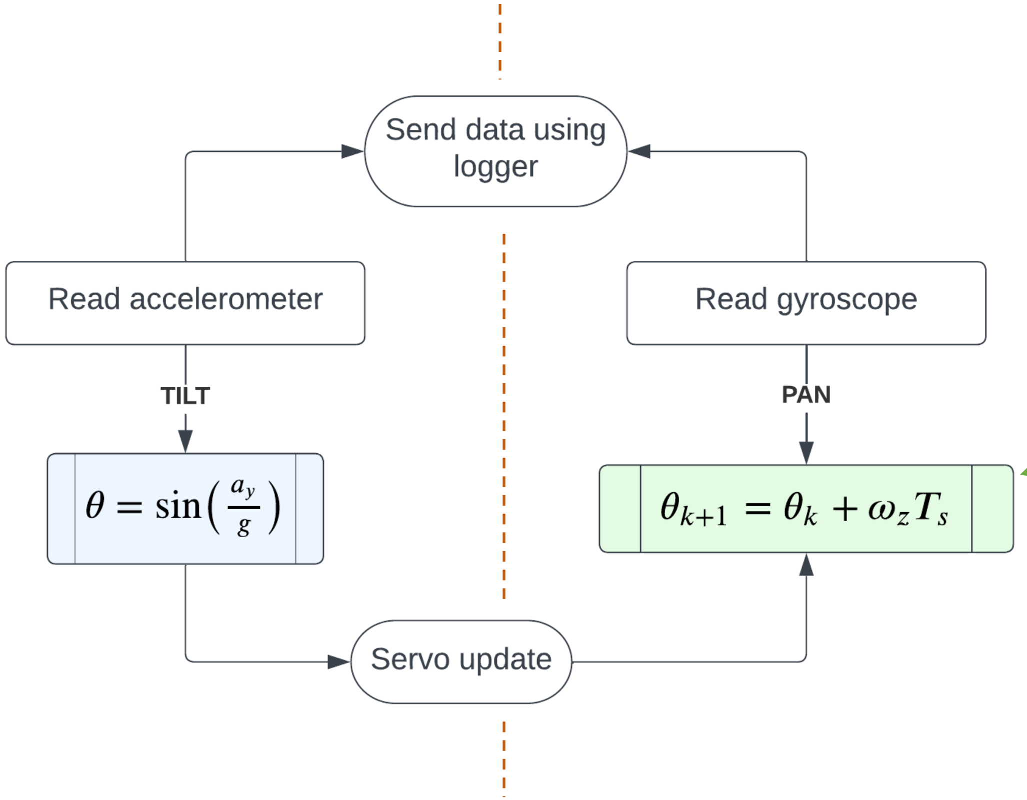


Control loop

The following controllers for PAN and TILT angle work independently since they act on different axes (the also rely on different measurements)

[**Ex1**] "Smooth" stabilization of the camera

[**Ex2**] Camera looking at a fixed point



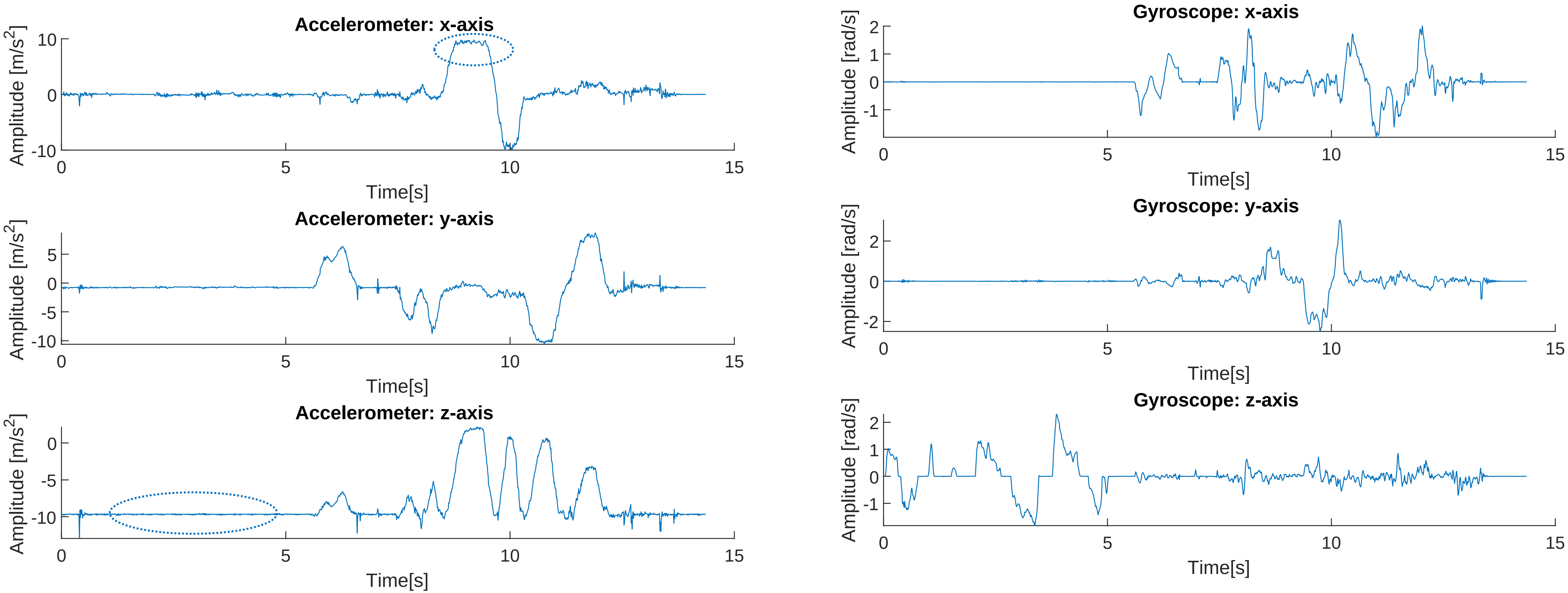
Numerical Integration

ω_z Angular speed

θ_k Angle at time k

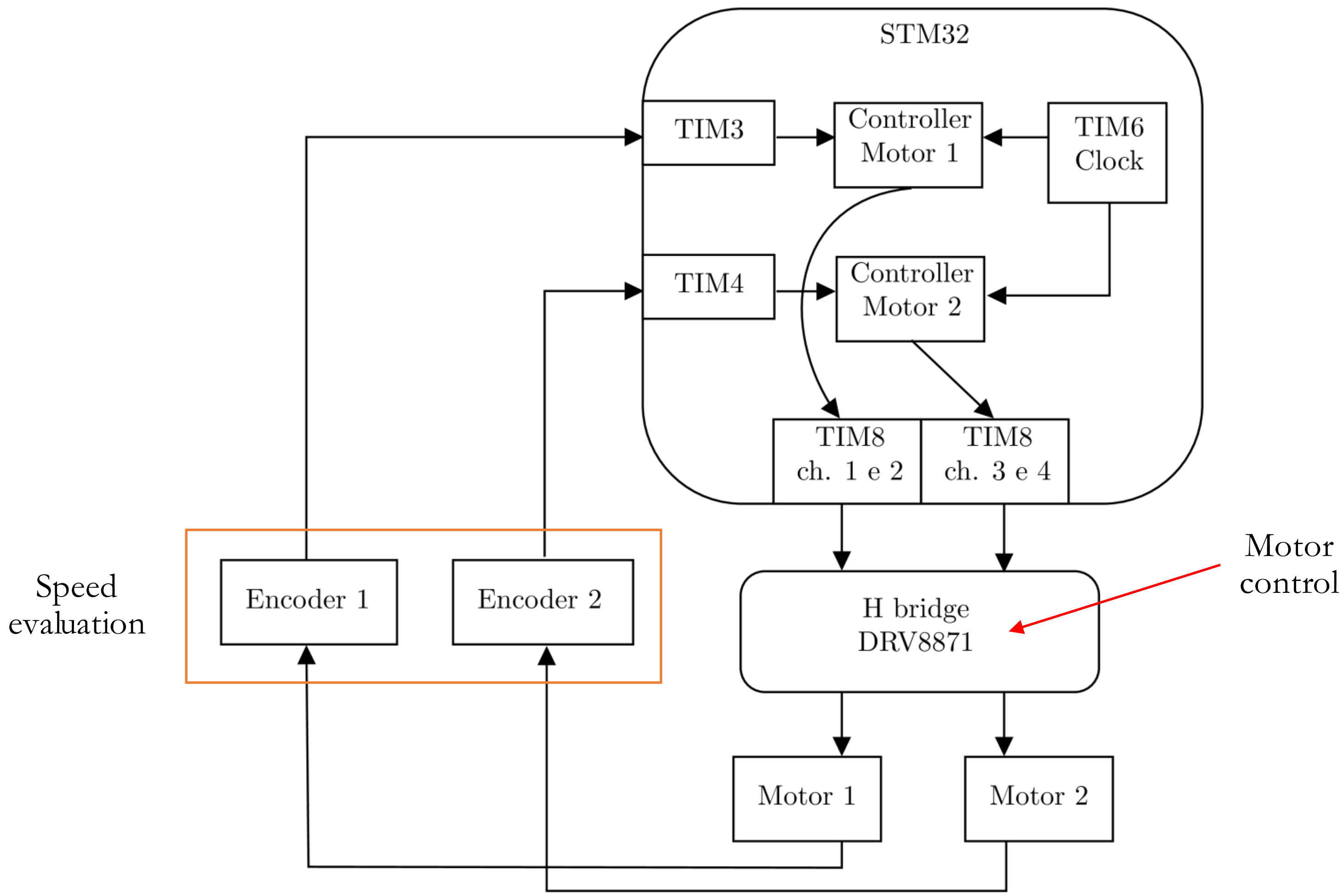
T_s Sampling time

Data logging



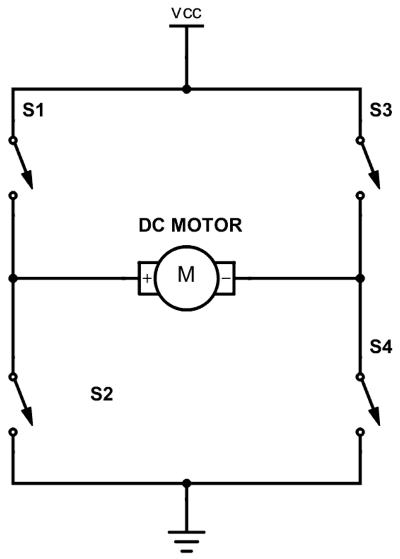
Axis perpendicular to the ground: 9.8 [m/s] (gravity acceleration)

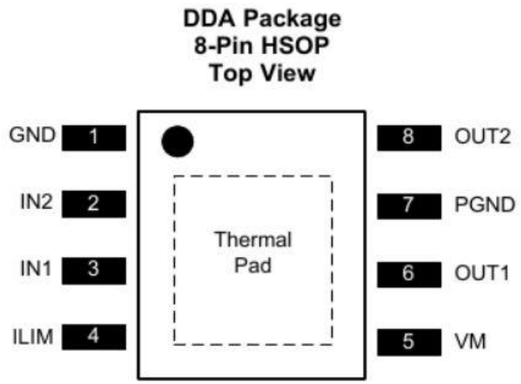
Laboratory 3 – Motor control



DRV8871 - Bridge control

DRV8871 is a brushed DC motor driver that can control a motor bidirectionally by implementing an **H-Bridge**. In order to use it we have to use *_HAL_TIM_SET_COMPARE()*

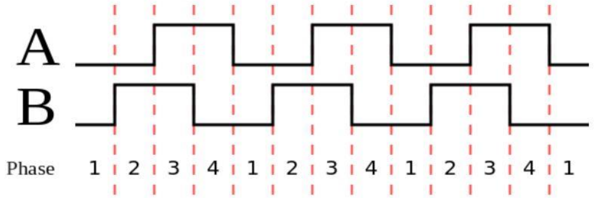




IN1	IN2	OUT1	OUT2	Mode name
0	0	Hi-z	Hi-z	*Coast*
0	1	L	H	*Reverse*
1	0	H	L	*Forward*
1	1	L	L	*Brake*

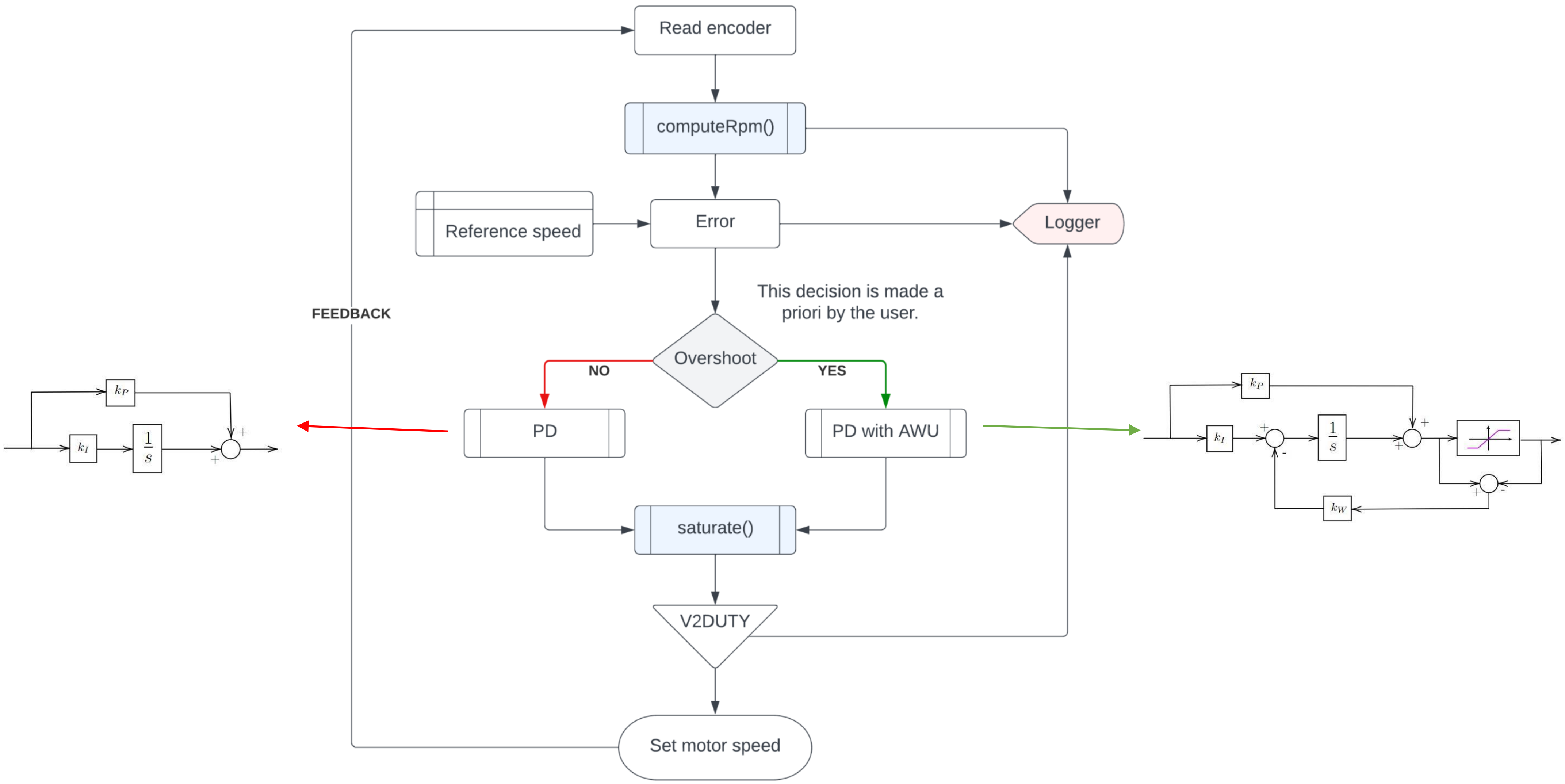
Quadrature encoder

It converts position information into an electrical signal. This encoder works in quadrature. Work in series with a filter to reduce noise

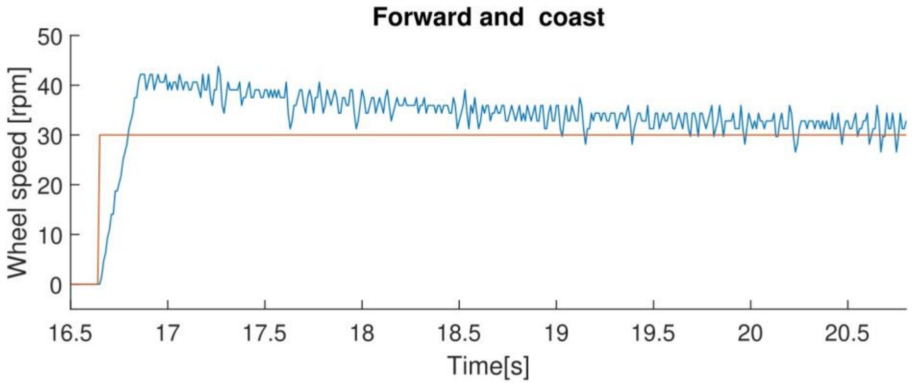


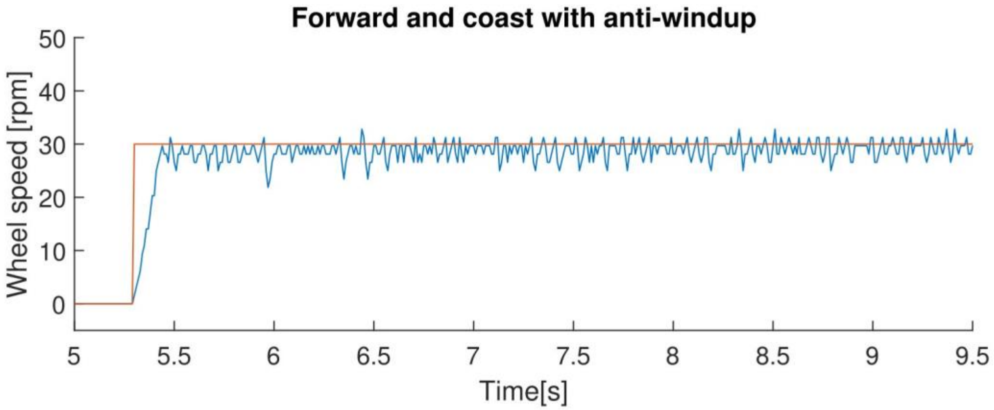
Flowchart

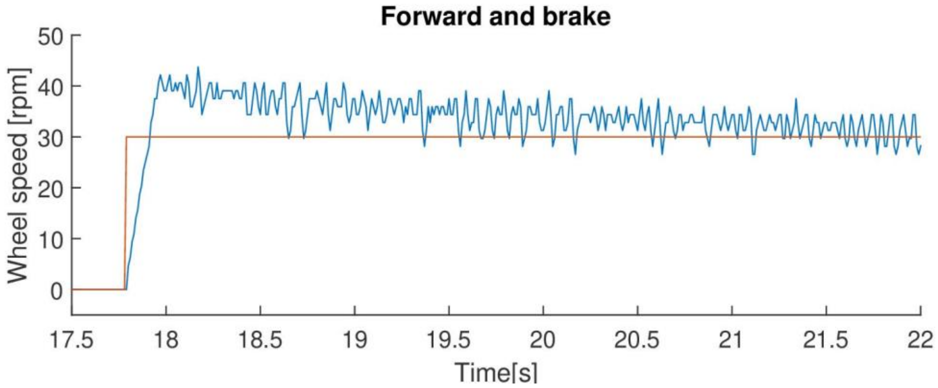
This scheme is the same for each motor, but they are indipendent

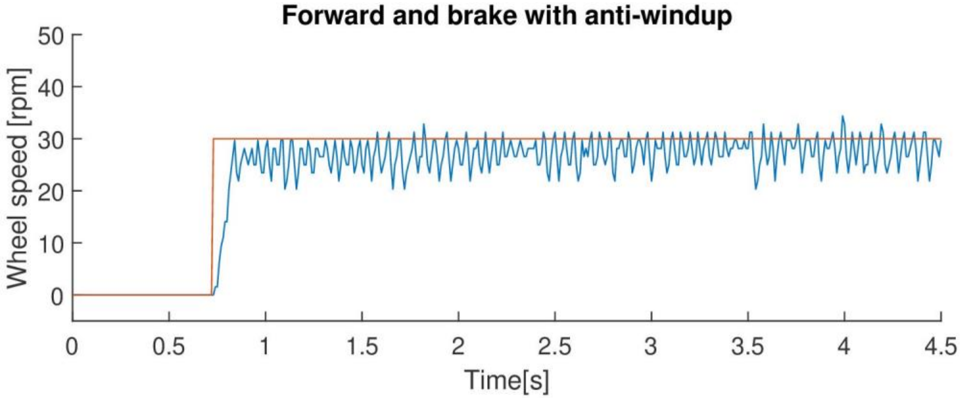


Results

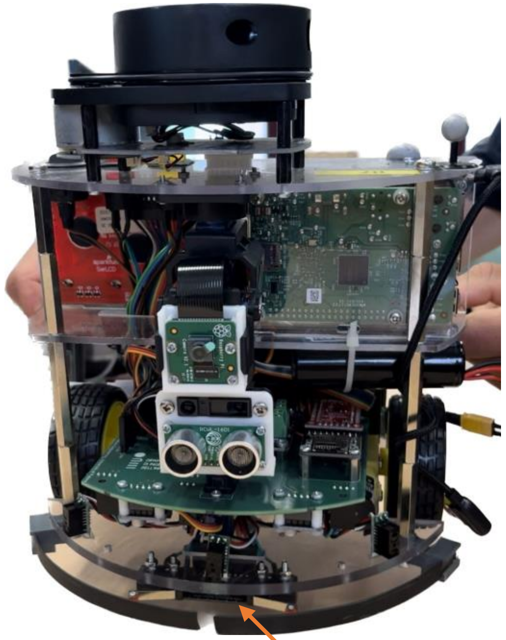

Forward and coast
Wheel speed [rpm]
Time[s]


Forward and coast with anti-windup
Wheel speed [rpm]
Time[s]

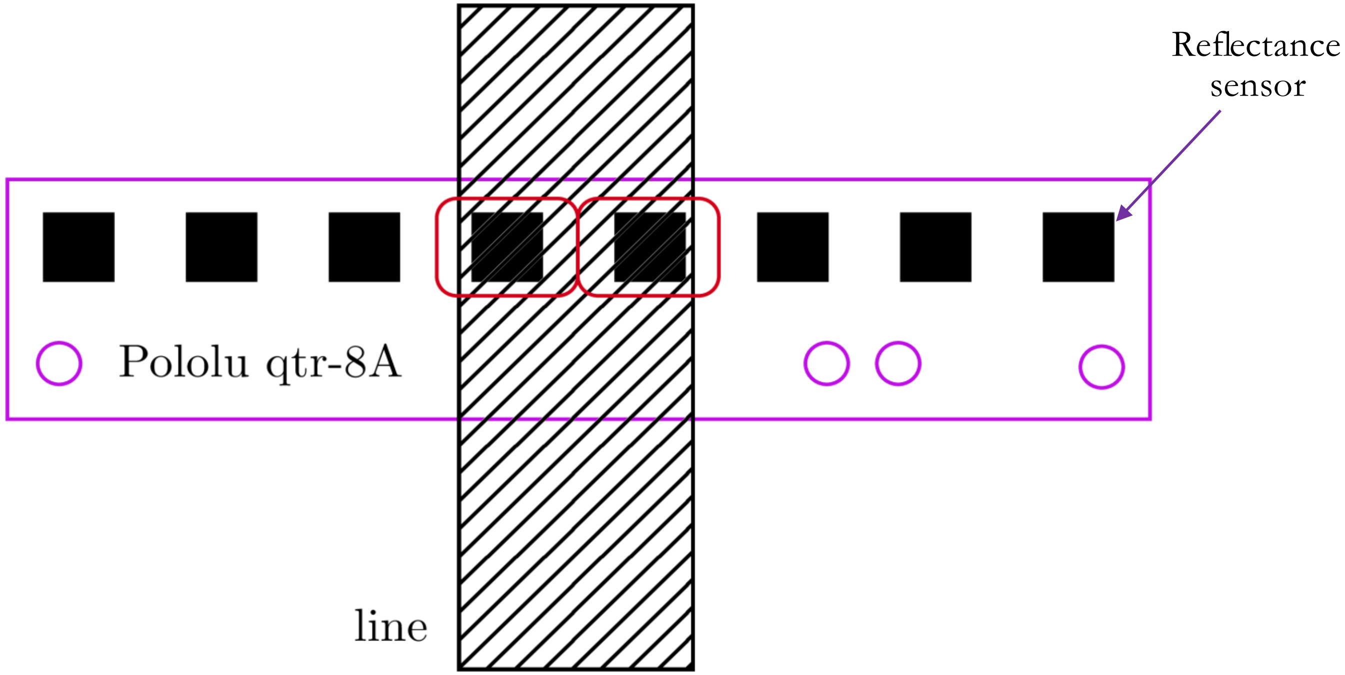

Forward and brake
Wheel speed [rpm]
Time[s]


Forward and brake with anti-windup
Wheel speed [rpm]
Time[s]

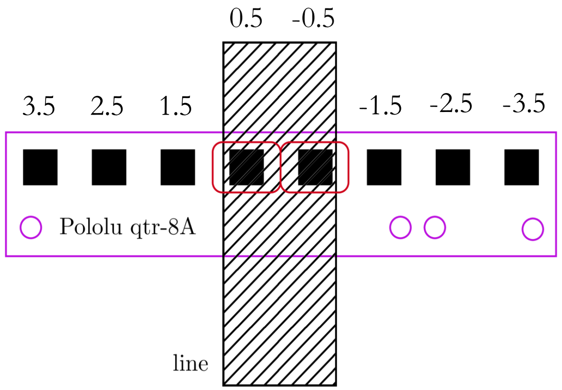
Laboratory 4 – Line tracker



Polulu reflectance sensor



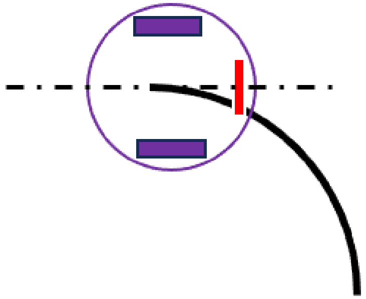
Control architecture

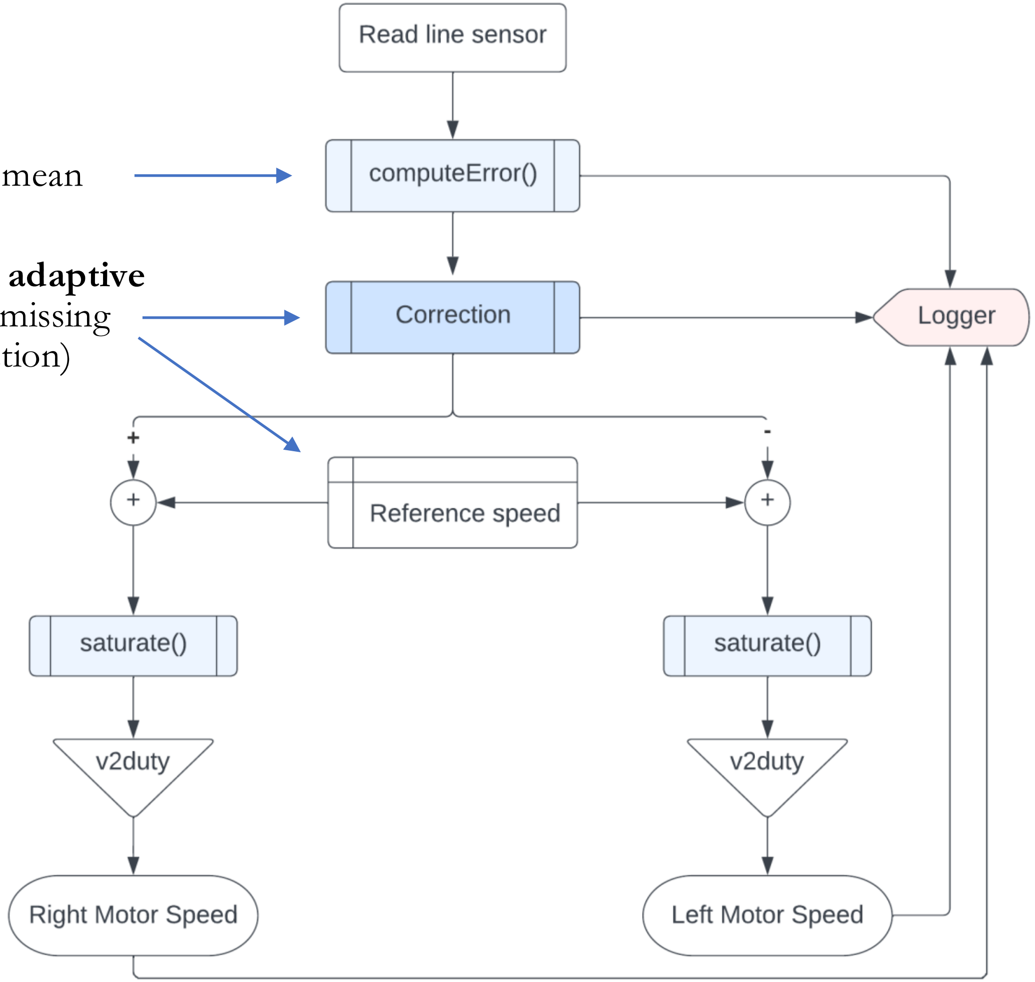


Weighted mean

We adopted an **adaptive** control law (missing integral action)

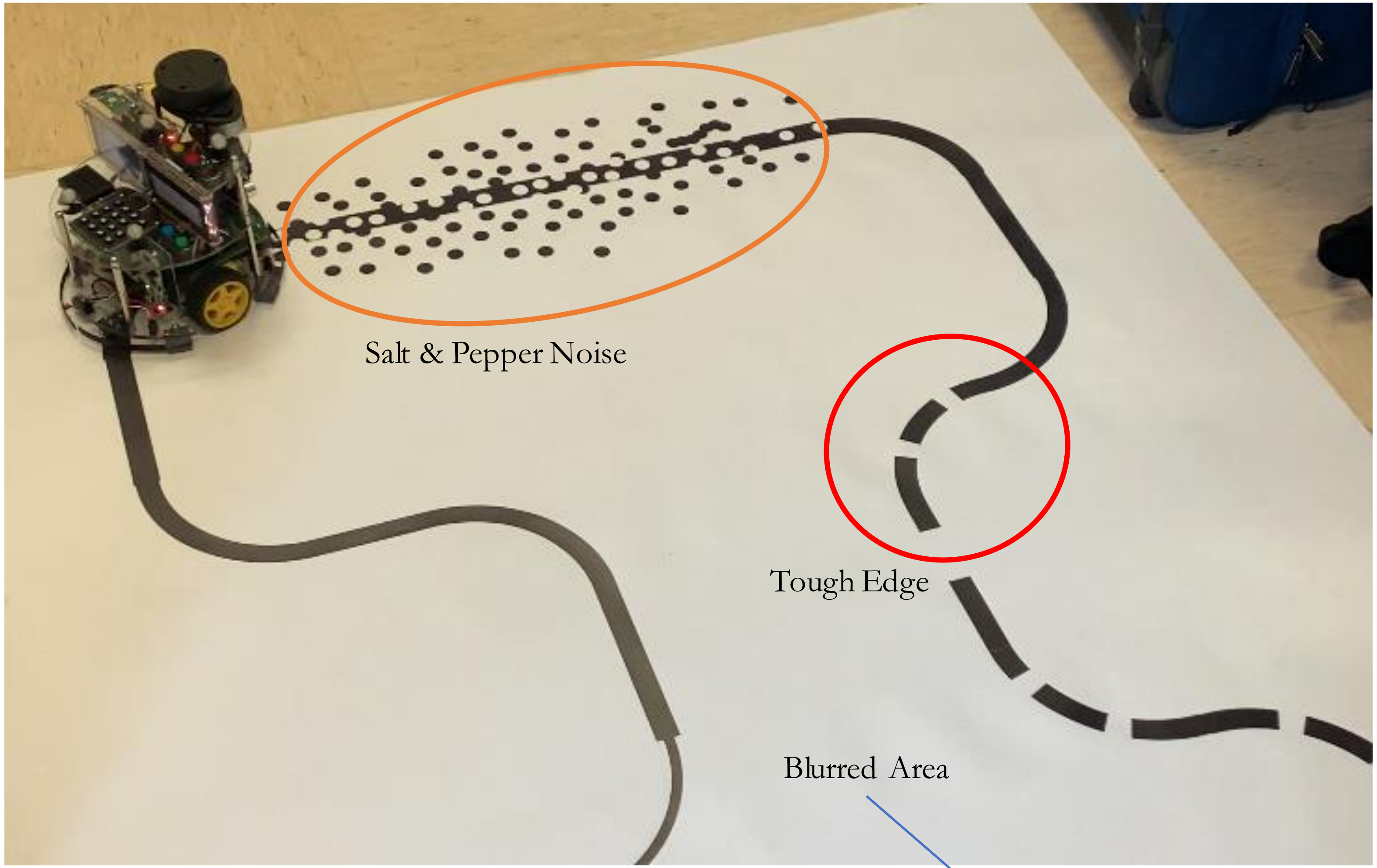
Limit edge and saturation value





It works through WIFI

Conclusions



FreeRTOS

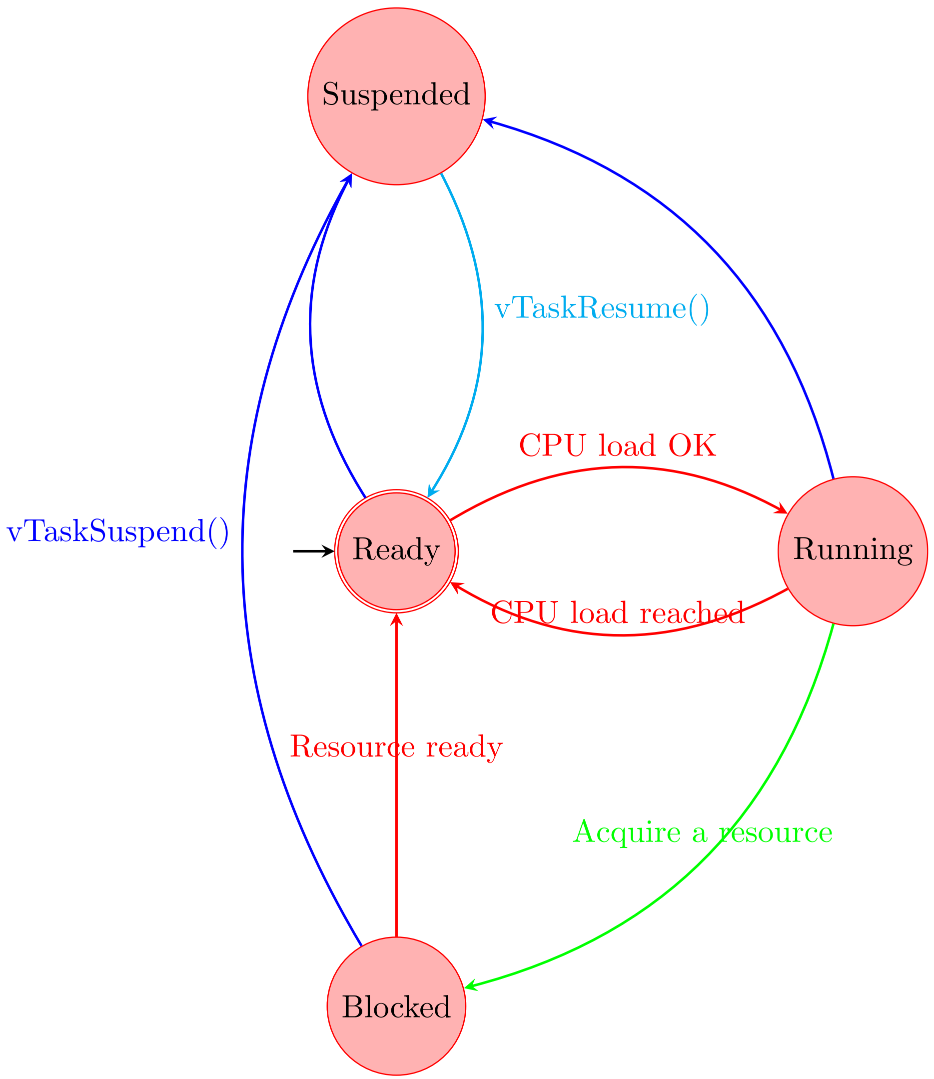
FreeRTOS (*Free Real Time Operating System*) is a real-time OS kernel for embedded device.

1. Simplify tasks scheduling (Lab1 Extra and Lab2 Ex2)
2. Enable Safer Inter-Task communications (Lab2 Ex3-4)

Tasks

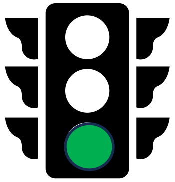
Each task is an execution unit with the following properties:

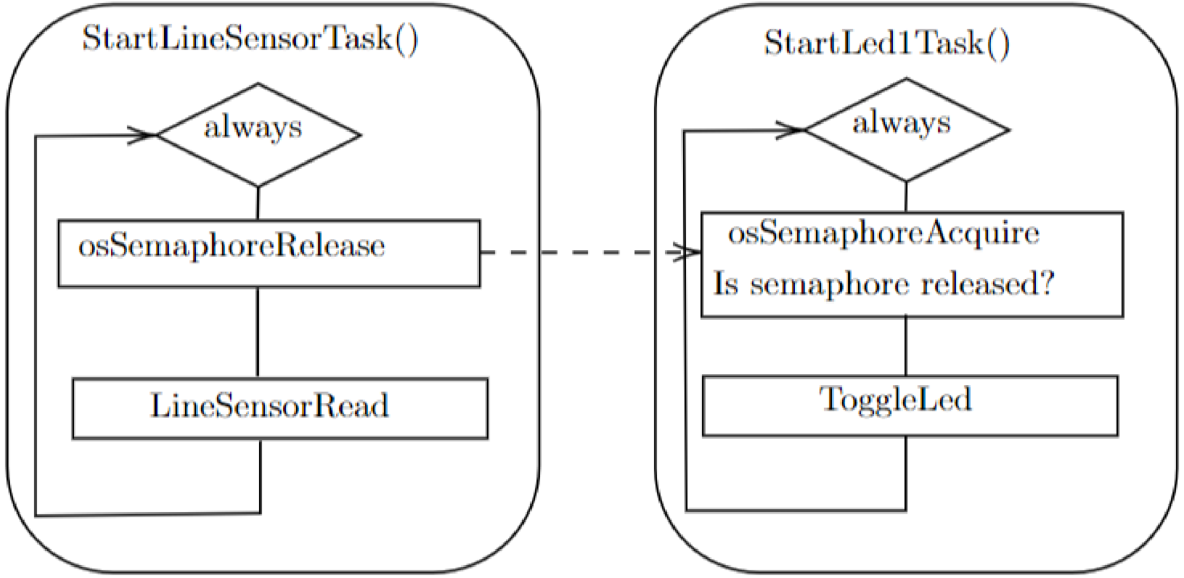
1. One task execute at a time;
2. No knowledge on scheduler activity;
3. Own stack to save exec. Content;
4. Can be prioritized;
5. Can be in one of the four states;



Laboratory 1 Extra – Task Scheduling

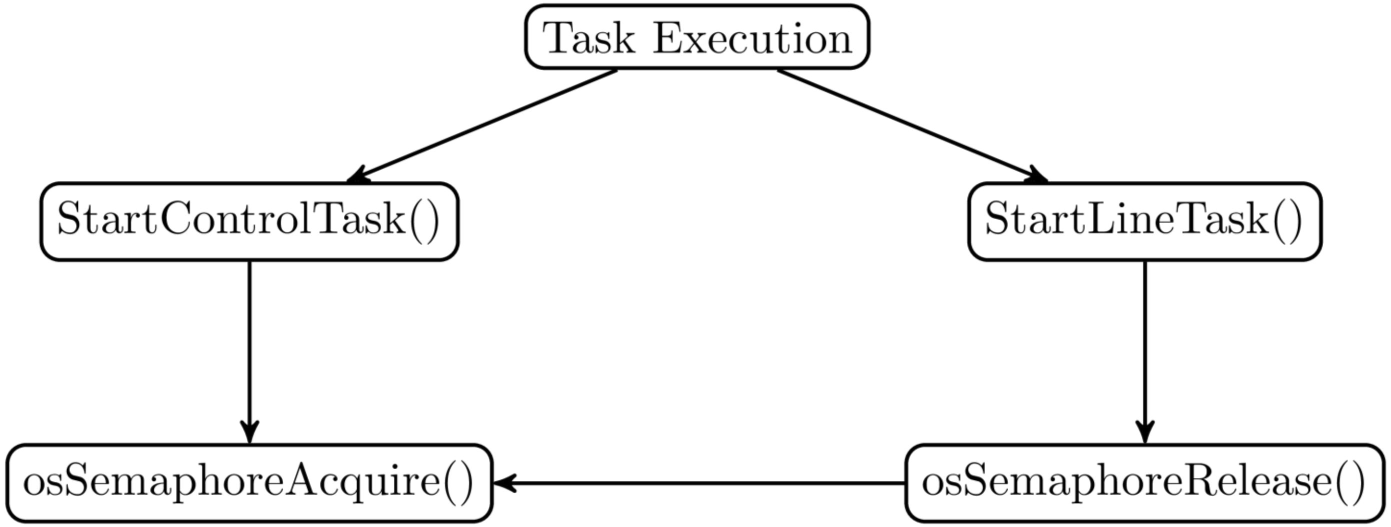
Semaphores are used to coordinate task execution





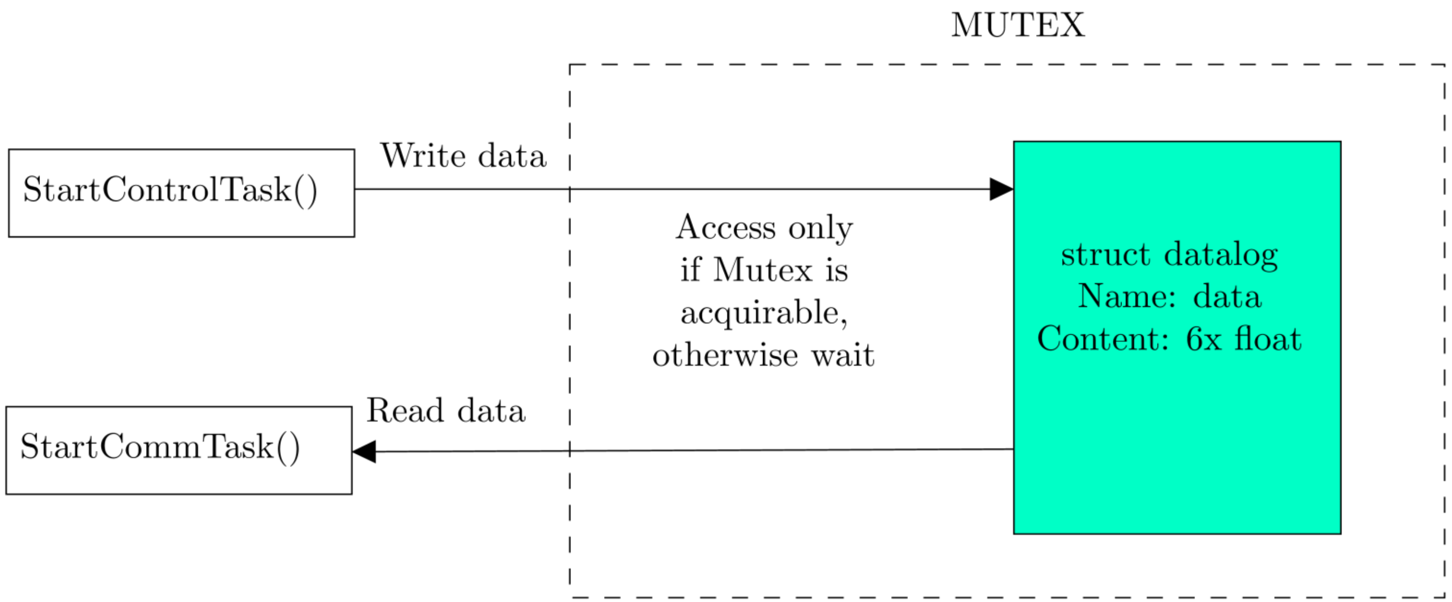
Laboratory 5 Ex 2 – Resource usage

Similar working principle as EX 1 Extra



Laboratory 5 Ex3-4 – Mutex and Queue

A task can access a critical region only if the mutex is locked (check if other tasks are already in the critical region)



Conclusions

- ✓ All the given tasks has been successfully developed and tested (all extra points too)
- ✓ Perfect camera stabilization
- ✓ Good PID performances (it could have been tuned better)
- ✓ Good performance obtained in line following, being able to complete the most complex circuit.
- ✓ Only Lab 5 ex 4 on queues has not been completed due to lack of time (program compiles but does not work correctly)