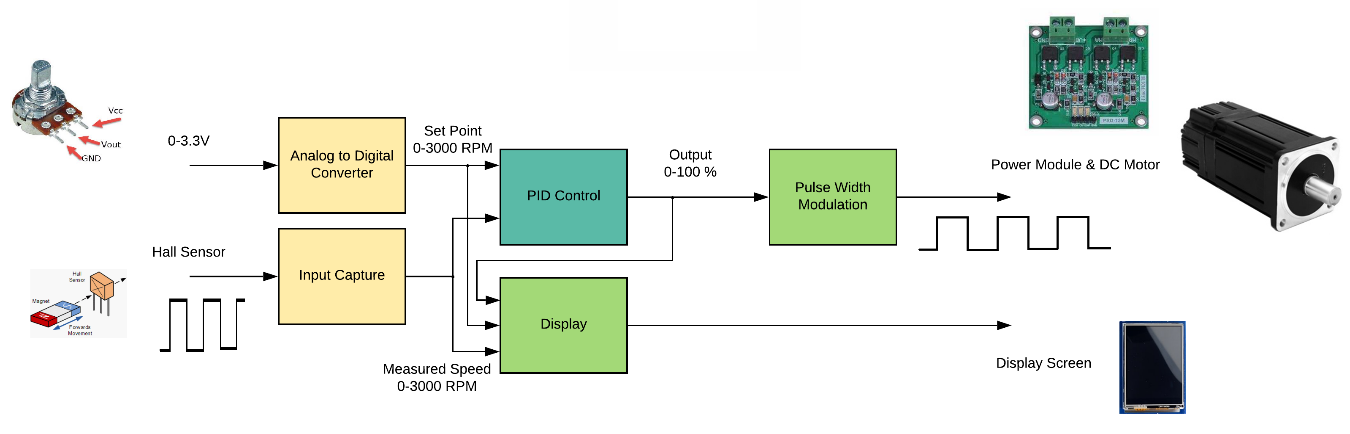
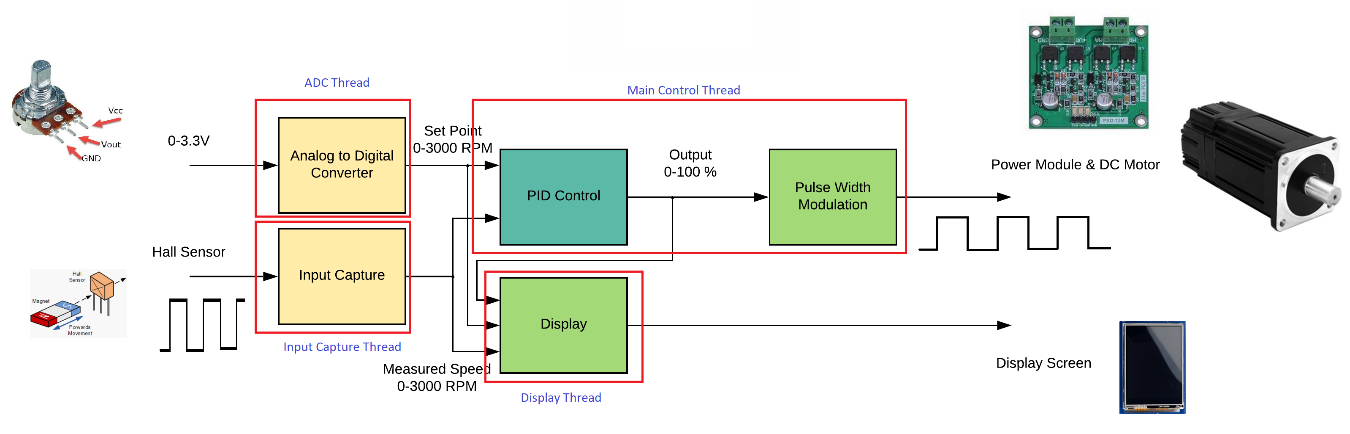
**Diagrama a bloques del sistema**



**Threads Implementation**

The structure of this project consists in four threads with the following names:

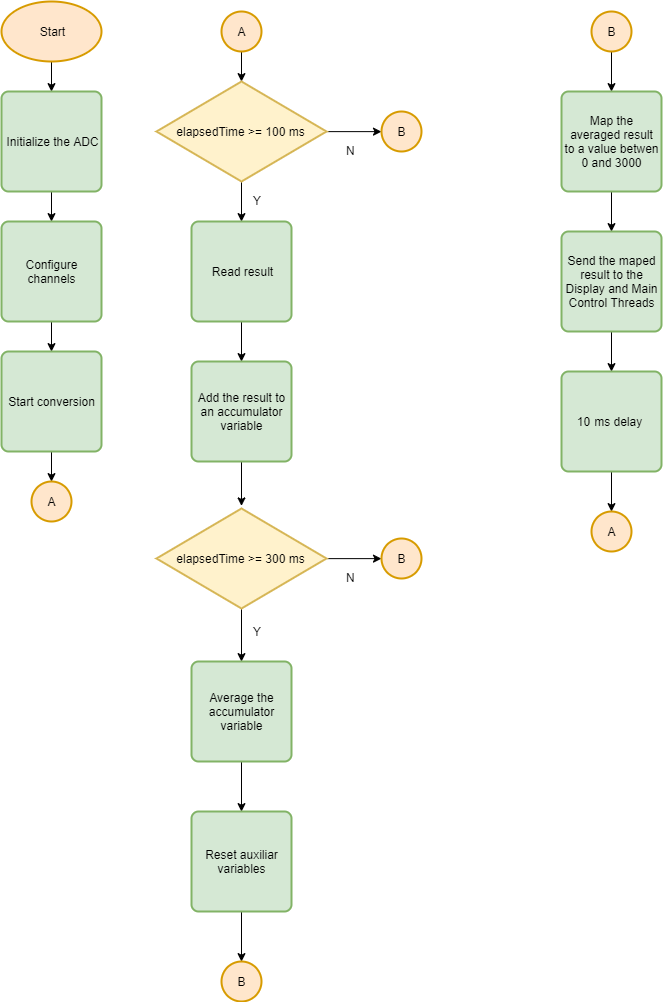
* ADC Thread
* Inputs: adc data (0 - 4095)
* Outpus: set point (0 - 3000 RPM)
* Input Capture Thread
* Inputs: measured period in ticks
* Outpus: measured speed (0 - 3000 RPM)
* Main Control Thread
* Inputs: set point (0 - 3000 RPM) and measured speed (0 - 3000 RPM)
* Outpus: PWM duty cycle (0 - 100 %)
* Display Thread
* Inputs: set point (0 - 3000 RPM), measured speed (0 - 3000 RPM) and duty cycle (0 - 100 %)
* Outpus: Display control signals and values to display.

**Analog to Digital Converter (ADC HAL Module)**

* Module Name: g\_adc0
* Resolution: 12 bits (0 - 4095)
* Mode: Continuous Scan
* Channel: 0
* Input Voltage Range: 0 - 3.3V

API functions used:

|  |  |
| --- | --- |
| **Function name** | **Example API Call and Description** |
| .open | g\_adc.p\_api->open(g\_adc.p\_ctrl, g\_adc.p\_cfg);  Initialize ADC Unit; apply power, set the operational mode, trigger sources, interrupt priority, and configurations common to all channels and sensors. |
| .scanCfg | g\_adc.p\_api->scanCfg(g\_adc.p\_ctrl, g\_adc.p\_channel\_cfg);  Configure the scan including the channels, groups and scan triggers to be used for the unit that was initialized in the open call. |
| .scanStart | g\_adc.p\_api->scanStart(g\_adc.p\_ctrl);  Start the scan (in case of software trigger) or enable the hardware trigger. |
| .read | g\_adc.p\_api->read(g\_adc.p\_ctrl, ADC\_REG\_CHANNEL\_13, &adc\_data);  Read ADC conversion result(s). |

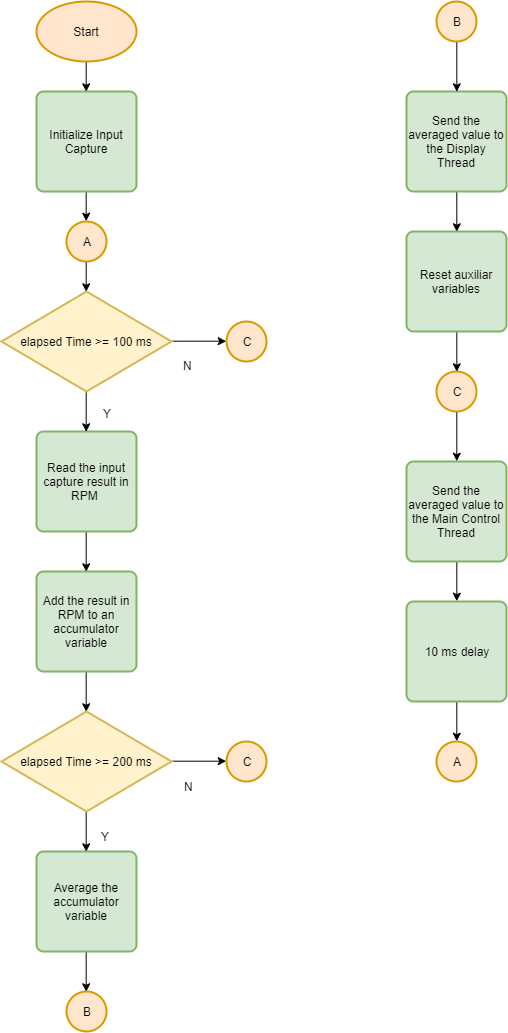


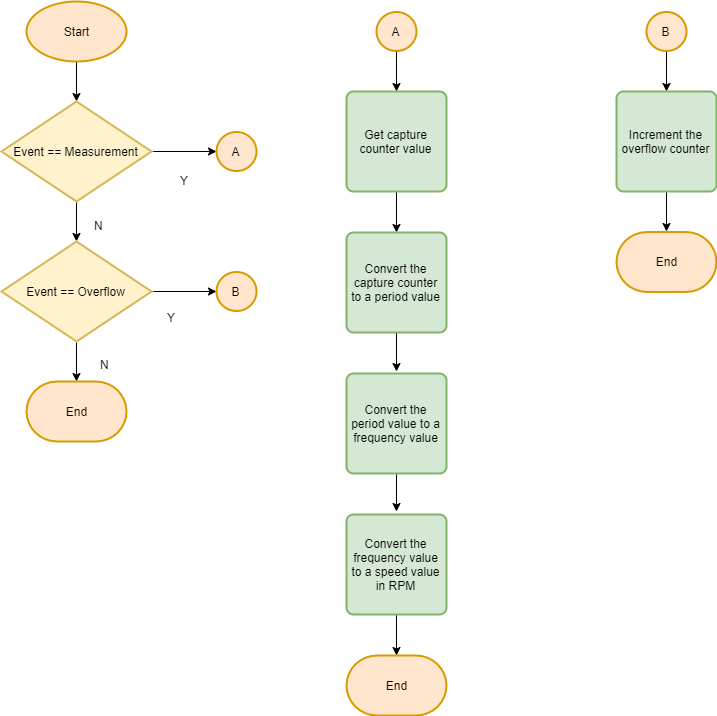
**Input Capture (Input Capture HAL Module)**

* Module Name: g\_input\_capture
* Mode: Period
* Signal Edge: Rising
* Repetition: Periodic
* Callback: InputCaptureCallback
* Channel: 2

API functions used:

|  |  |
| --- | --- |
| **Function name** | **Example API Call and Description** |
| .open | g\_input\_capture.p\_api->open(g\_input\_capture.p\_ctrl, g\_input\_capture.p\_cfg);  Opens the Input Capture HAL and initializes configuration. |





**Pulse Width Modulation (GPT HAL Module)**

* Module Name: g\_timer1
* Mode: PWM
* Period Value: 1ms
* Duty Cycle Unit: Percent
* Channel: 1

API functions used:

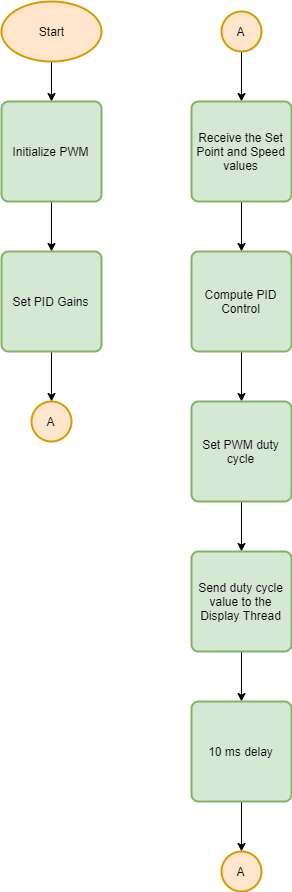
|  |  |
| --- | --- |
| **Function name** | **Example API Call and Description** |
| .open | g\_timer0.p\_api->open(g\_timer0.p\_ctrl, g\_timer0.p\_cfg)  Initial configuration. |
| .start | g\_timer0.p\_api->start(g\_timer0.p\_ctrl)  Start the counter. |
| .dutyCycleSet | g\_timer0.p\_api->dutyCycleSet(g\_timer0.p\_ctrl, period, unit, pin)  Sets the time until the duty cycle expires. |

Initialize PWM

Read Value from ADC

Adjust duty cycle with the value From ADC

**Main Control Thread**



**PID Controller Pseudocode**

previous\_error = 0

integral = 0

loop:

error = setpoint - measured\_value

integral = integral + error \* dt

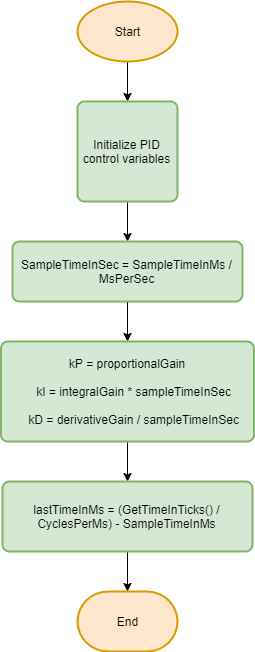
derivative = (error - previous\_error) / dt

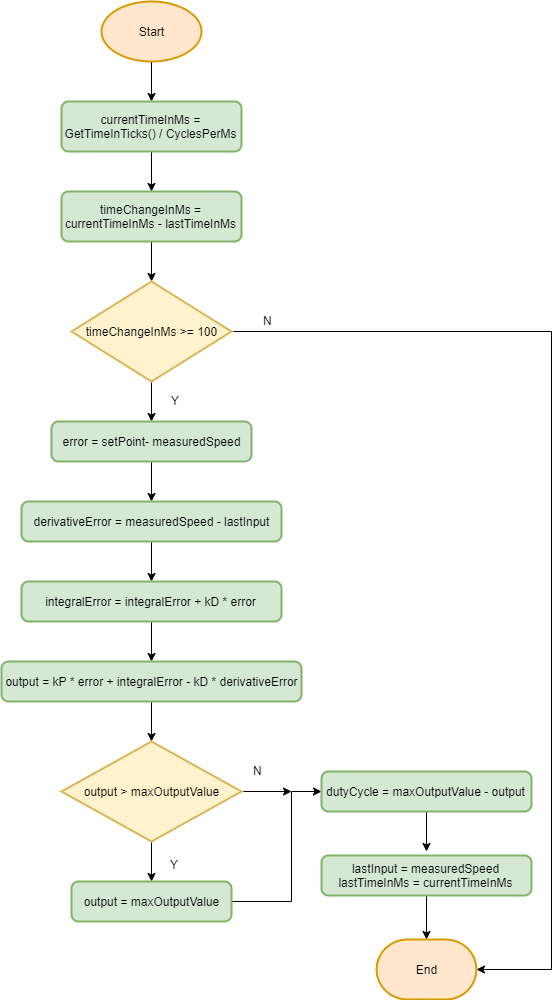
output = Kp \* error + Ki \* integral + Kd \* derivative

previous\_error = error

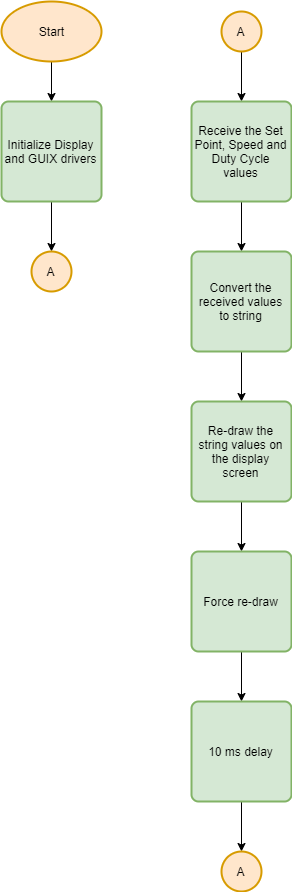
wait(dt)

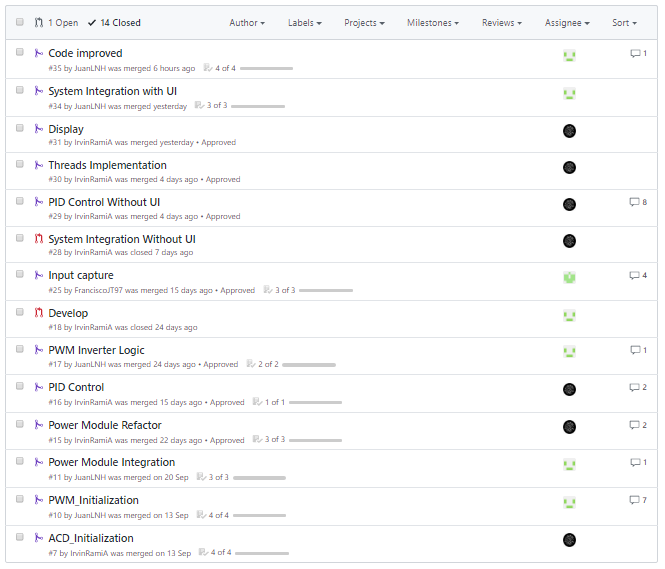
goto loop





**Display Thread**





**UML Diagrams**

