Table of contents

1. Project

Object Tracking with a Radio Controlled Quadcopter

2. Members

- Julião Gessé Fernandes

3. About the project

The goal of this project is to implement a simple object tracking oriented by a predefinide color using the images captured by a USB camera onboard a quadcopter. The main ideia is to use a *Intel Edison* (***Figure 1***) board to do the image processing and to command the movimentation of the quadcopter when is necessary (general commands will received by a commercial radio control). The quadcopter will use the *MultiWii* controller as stabilization system (***Figure 2***). In the ***Figure 3*** you can see a high level block diagram of the project.

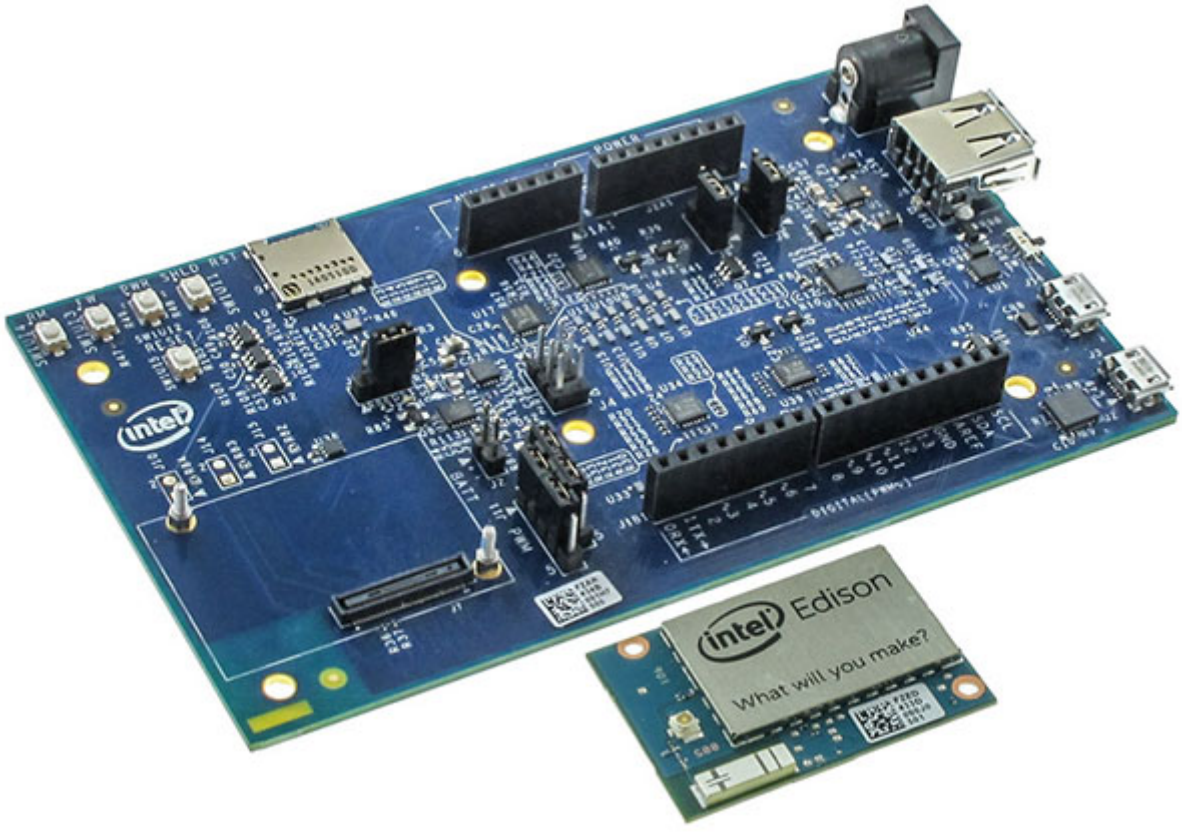


Figure 1 - Intel Edison and the official Arduino Compatible Board

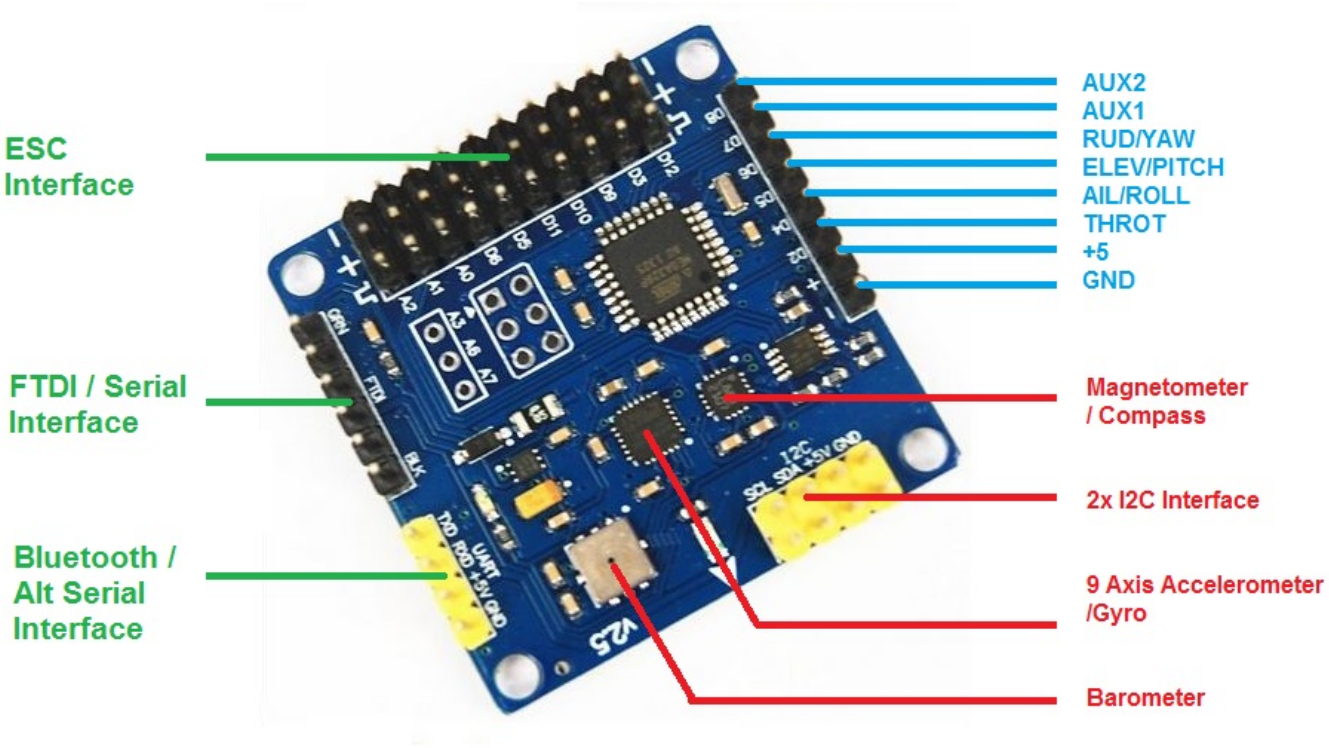


Figure 2 - MultiWii SE v2.5 Controller

The quadcopter will be controlled mainly by the radio control, but at the same time, the tracking of an object can be activated via a switch located at the radio control. The activation of this switch will be acknowledged by the receiver, that will enable the ***en_track*** signal (*ADC* input) at the *Intel Edison* board. Starting at this moment, the object tracking system at the *Intel Edison* board will seek a target of predefined color in the current image captured by the camera. When a target have been acquired, the track system will work to keep the target always in the center of the camera frame, this will be done sending movimentation commands to the quadcopter through the *MultiWii* serial interface, using the *MultiWii Serial Protocol* (MSP).

In this object tracking project, the camera will be installed under the quadcopter, capturing images of the ground, so it can move left / right and forward / backward, maintaining a constant height.

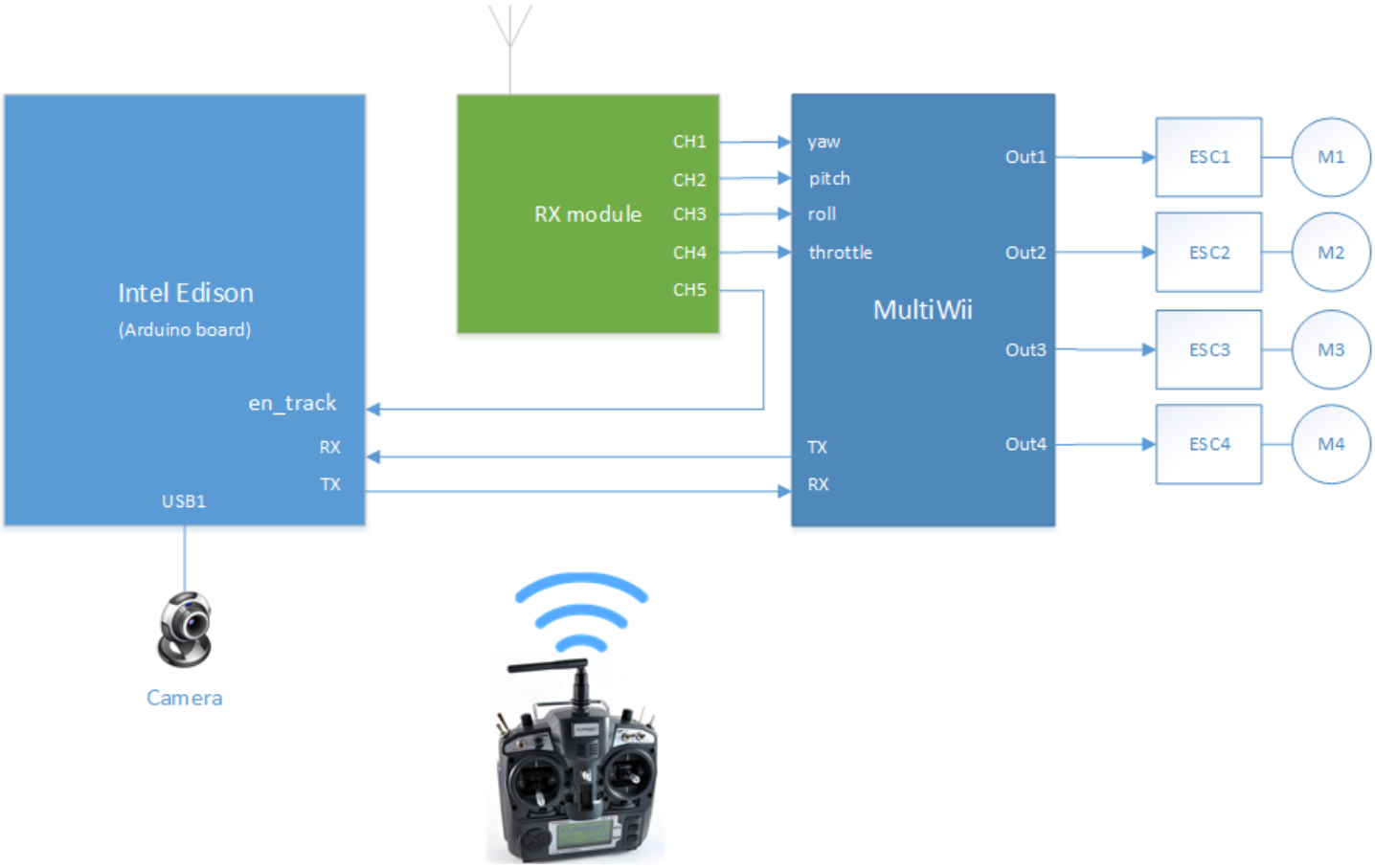


Figure 3 - Blocks diagram

The movimentation of the quadcopter is done by changing the throttle applied to each one of the four brushless motors, the throttle to each motor should be adequated for the desired movement, like you can see in the ***Figure 4***. This will be managed but the *MultiWii* controller board.

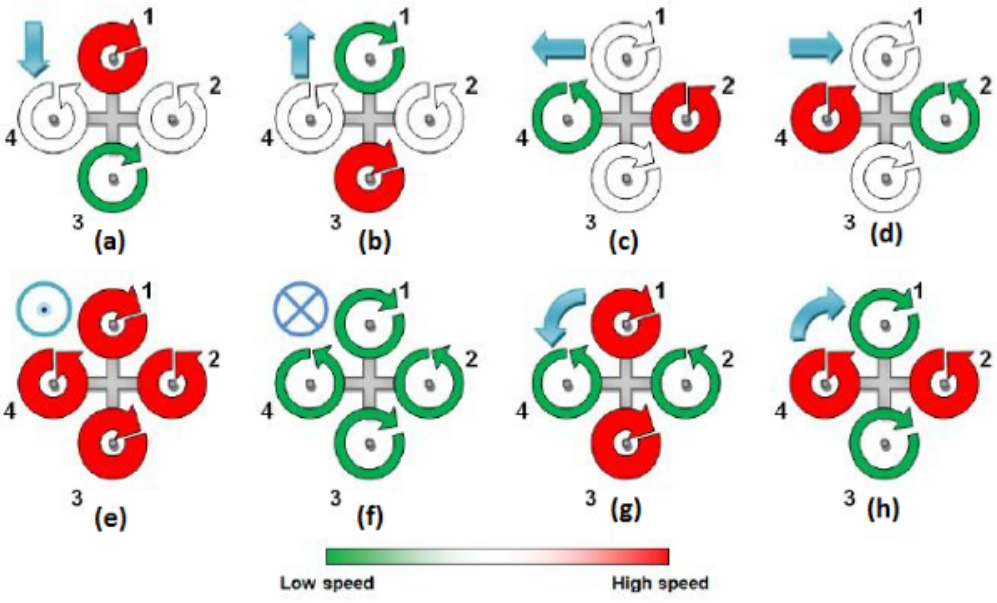


Figure 4 - Quadcopter states ("+" configuration). (**a**) Forward motion; (**b**) backwards motion; (**c**) movement left; (**d**) movement right; (**e**) increase altitude; (**f**) decrease altitude; (**g**) leftwards rotation; (**h**) rightwards rotation.

4. Requirements

- Radio Controller (TX + RX) (Borrowed by LISHA)
- MultiWii board (Borrowed by LISHA)
- Quadcopter frame with brushless motors and ESCs (Borrowed by LISHA)
- USB camera (**To be acquired** or borrowed by LISHA)
- LiPo battery 11.1 Volts >25C >3000mAh (**To be acquired** or borrowed by LISHA)

Utilized components details:

- 4x Brushless motor A2830-11 1000KV RCTimer Outrunner
- 4x ESC SK-30A Simonk Firmware ESC BEC 5V/2A 2-4S LiPo RCTimer
- 4x Propeller 10x4.5 (two reversed)

4.1 Non-Functional Requirements

- The Quadcopter's battery should provide enough power for the motors, MultiWii and the Intel Edison board.
- The battery should supply enough power for the Quadcopter for at least 10 minutes and save power for landing if the battery is low (disabling the image processing function).
- The camera images should have the adequate resolution to be processed in the Intel Edison board (and/or the images could be optimized for performance by downscaling resolution if necessary).
- The processing time between a camera frame and another should be of the order of milliseconds.
- The response time to data received from MultiWii by Intel Edison board must be of the order of milliseconds.
- The Quadcopter must do smooth movements, because the camera is fixed (Note: Ideally, the camera should be installed on a gimbal, so the inclination caused by the movement of the Quadcopter is compensated and the camera keep the target on the frame).

4.1.1 Battery autonomy analysis

Components weight:

- 620g Quadcopter frame + 4 ESCs + 4 brushless motors
- 66g 4 Propeller with spinner (16,5g each)
- 11g MultiWii board
- 22g RX module
- 7g Intel Edison module
- 46g Intel Edison Arduino board
- 32g USB Camera

- 193g Battery 2800mAh 3S 35C
- 361g Battery 5000mAh 3S 30C

- Just Intel Edison Kit + USB Camera: 85g

- Quadcopter without battery: 804g
- Quadcopter with 2800mAh battery: 997g
- Quadcopter with 5000mAh battery: 1165g

all data without guarantee - Accuracy: +/-15%

xcopterCalc - Multicopter Calculator

News | Help | Tutorial | Language: english

General	Motor Cooling: good	# of Rotors: 4 flat	Model Weight: 804 g 28.4 oz	less Battery	Frame Size: 450 mm 17.72 inch	FCU Tilt Limit: no limit	Field Elevation 500 m ASL 1640 ft ASL	Air Temperature 25 °C 77 °F	Pressure (QNH): 1013 hPa 29.91 inHg	
Battery Cell	Type (Cont. / max. C) - charge state: LiPo 2500mAh - 20/30C - full	Configuration: 3 S 1 P	Cell Capacity: 2500 mAh 2500 mAh total	max. discharge: 85%	Resistance: 0.009 Ohm	Voltage: 3.7 V	C-Rate: 20 C cont. 30 C max	Weight: 193 g 6.8 oz		
Controller	Type: max 30A	Current: 30 A cont. 30 A max	Resistance: 0.008 Ohm	Weight: 40 g 1.4 oz	**Accessories**	Current drain: 0 A	Weight: 0 g 0 oz			
Motor	Manufacturer - Type (Kv): RCTimer A2830-11 (1000) search... Prop-Kv-Wizard	KV (w/o torque): 1000 rpm/V	no-load Current: 0.7 A @ 10 V	Limit (up to 15s): 210 W	Resistance: 0.127 Ohm	Case Length: 30 mm 1.18 inch	# mag. Poles: 14	Weight: 52 g 1.8 oz		
Propeller	Type - yoke twist: custom - 0°	Diameter: 10 inch 254 mm	Pitch: 4.5 inch 114.3 mm	# Blades: 2	PConst / TConst: 1.06 / 1.0	Gear Ratio: 1 : 1	Hack!	calculate		

Load: | Hover Flight Time: | electric Power: | est. Temperature: | Thrust-Weight: | specific Thrust:

Remarks:

Battery			**Motor @ Optimum Efficiency**			**Motor @ Maximum**			**Motor @ Hover**			**Total Drive**			**Multicopter**		
Load:	20.99	C	Current:	7.40	A	Current:	13.12	A	Current:	4.03	A	Drive Weight:	1042	g	All-up Weight:	1383	g
Voltage:	10.35	V	Voltage:	10.91	V	Voltage:	10.24	V	Voltage:	11.30	V		36.8	oz		48.8	oz
Rated Voltage:	11.10	V	Revolutions*:	9887	rpm	Revolutions*:	8435	rpm	Revolutions*:	5237	rpm	Thrust-Weight:	2.0	: 1	add. Payload:	1007	g
Energy:	27.75	Wh	electric Power:	80.7	W	electric Power:	134.4	W	Throttle (log):	46	%	Current @ Hover:	16.12	A		35.5	oz
Total Capacity:	2500	mAh	mech. Power:	66.2	W	mech. Power:	105.3	W	Throttle (linear):	55	%	P(in) @ Hover:	189.6	W	max Tilt:	55	°
Used Capacity:	2125	mAh	Efficiency:	82.1	%	Power-Weight:	388.6	W/kg	electric Power:	45.5	W	P(out) @ Hover:	141.8	W	max. Speed:	38	km/h
min. Flight Time:	2.4	min					176.3	W/lb	mech. Power:	35.5	W	Efficiency @ Hover:	74.8	%		23.6	mph
Mixed Flight Time:	5.8	min				Efficiency:	78.4	%	Power-Weight:	137.1	W/kg	Current @ max:	52.47	A	est. rate of climb:	6.1	m/s
Hover Flight Time:	7.9	min				est. Temperature:	38	°C		62.2	W/lb	P(in) @ max:	617.3	W		1201	ft/min
Weight:	579	g					100	°F	Efficiency:	77.9	%	P(out) @ max:	421.3	W	with Rotor fail:		
	20.4	oz							est. Temperature:	29	°C	Efficiency @ max:	68.3	%			
										84	°F						
									specific Thrust:	7.59	g/W						

5. Model

A simplified functional model was created in the Ptolemy II software, which can be seen in the ***Figure 5***.This model simulates the object tracking and the quadcopter position correction. I used a series of images simulating the frames captured by a camera (you can see the frames sequence in the ***Figure 7***), which are processed by an external program (described in the subsection 5.1). This program processes the frames, seeking for a target, the target is defined by a color range in the *HSV* color space. When a target is found, the movimentation command and the offset corrections for the X and Y coordinates are generated, this data would be used by the tracking controller (*Intel Edison*) for repositioning the quadcopter.

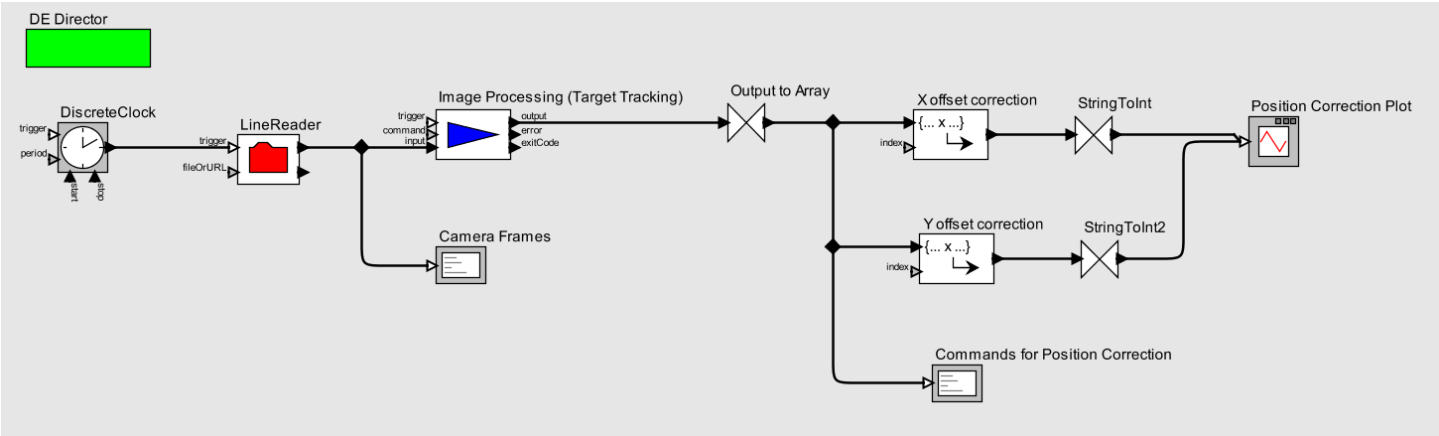


Figure 5 - Simulation Model

The simulation result can be seen in the ***Figure 6***, the image processing program works to centralize the tracked object in the center of the camera frame, generating the corrections offsets and the movimentation commands.

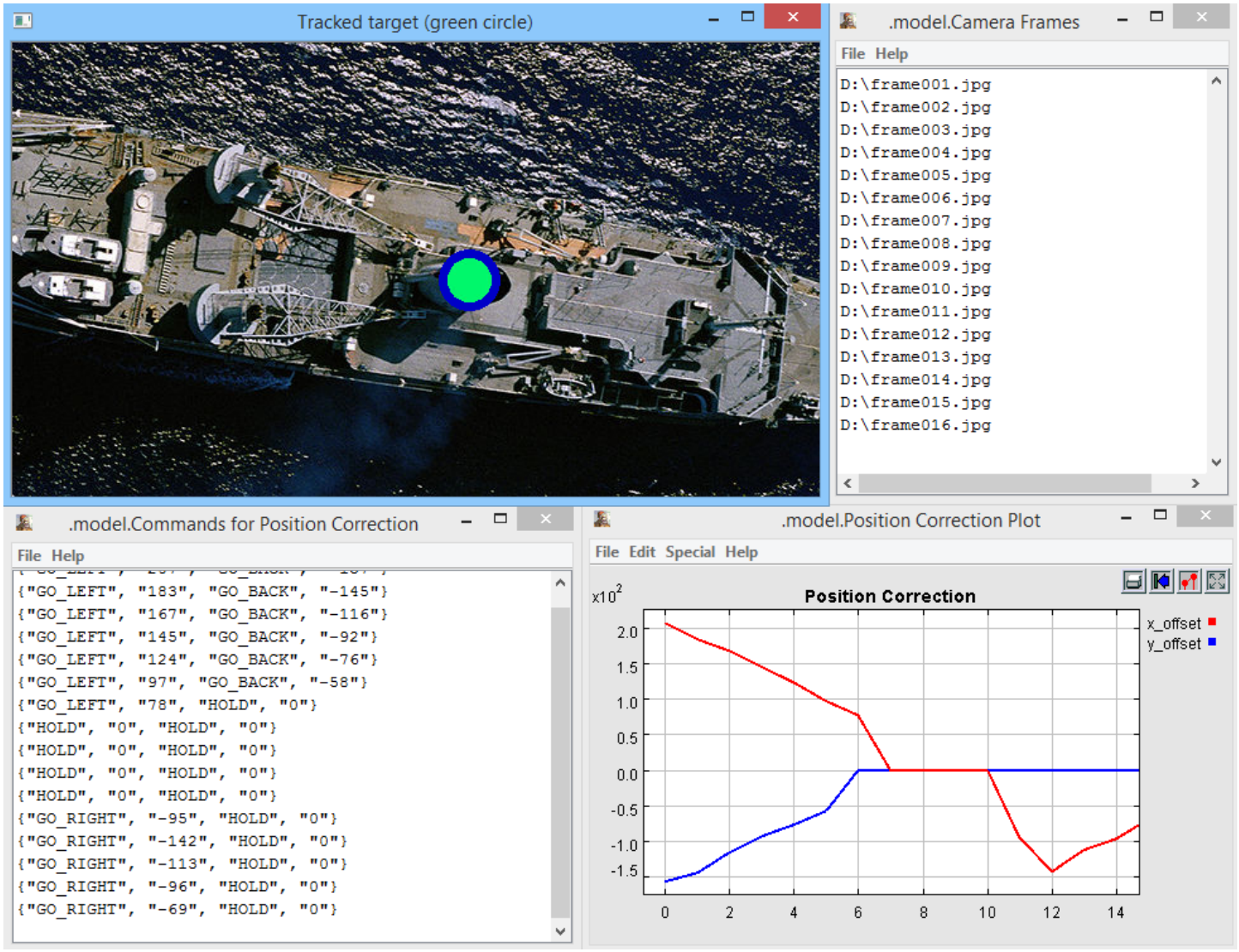


Figure 6 - Ptolemy II simulation results

The frames sequence in the ***Figure 7***, tries to simulate the desired behavior of the quadcopter in the real world, also considering the possibility of a rapid movement of the target (middle of the frames, when the quadcopter is on HOLD and the boat seems to move ahead).

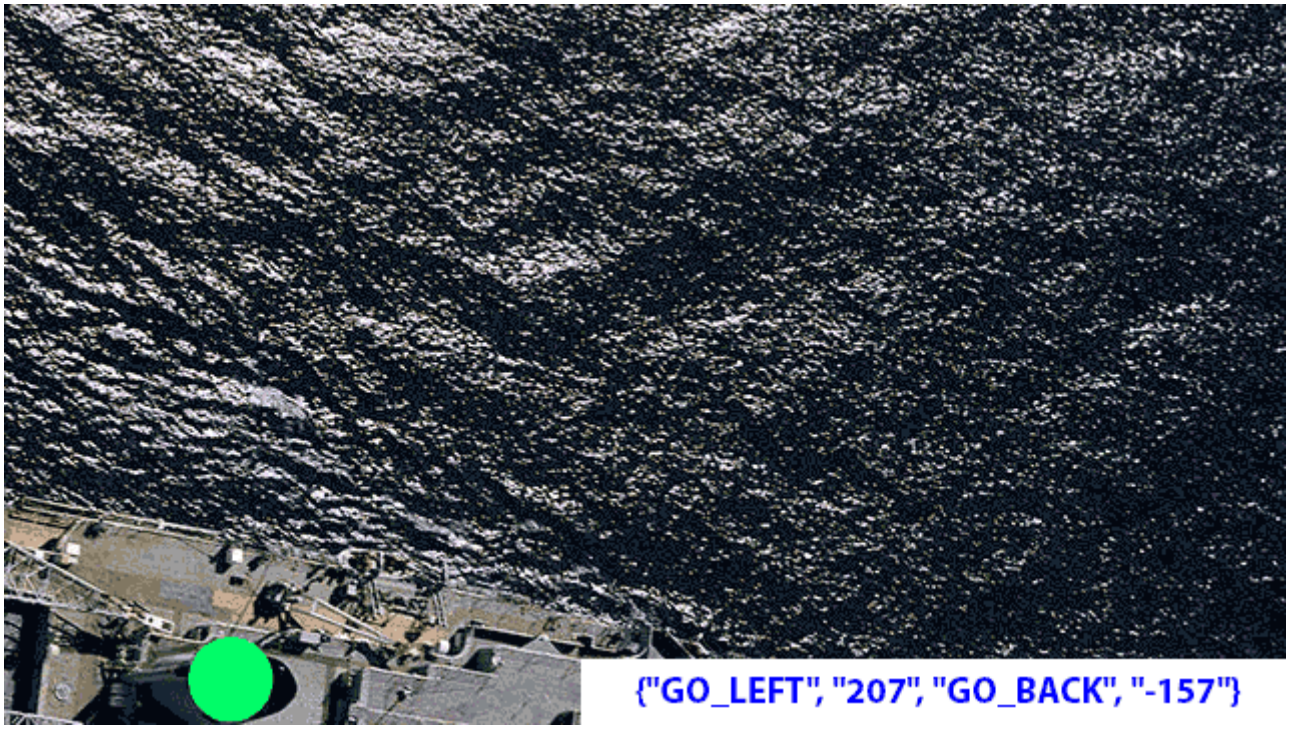


Figure 7 - Simulation camera frames and correction commands

5.1 Image processing (Program)

The program uses the *OpenCV* library to do the image processing. In this simplified model, it is not doing the tracking of objects properly, because each frame is processed independently, so, if there is more than one object in the scene with the same color, only the first to be located will be tracked, even if the first is not the object which was first in the previous frame. The source code can be seen here: `main.cpp`

6. Model of Computation

This model uses the Synchronous Dataflow paradigm model, since there is a communication between some components using messages across FIFO queues and there is a constraint in the time that some components can wait for receive a message while reading from the FIFO queue.

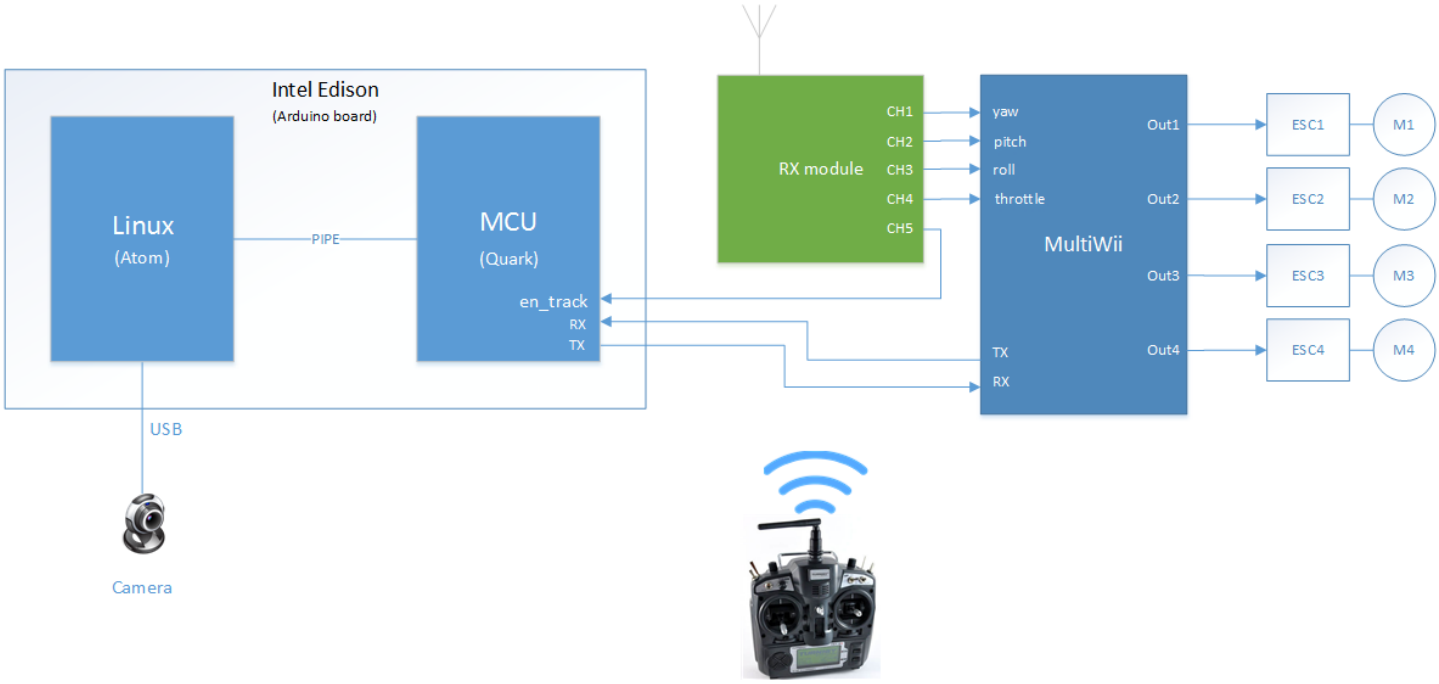
In this model we have two parts that do:

Process 1: Image processing, object tracking command based and telemetry data processing.
Process 2: MultiWii commander and telemetry collector.

Each of these parts run in a separated physical processor in the *Intel Edison* board, the **Process 1** run in the *Intel Atom* processor, inside the Linux host. And the **Process 2** run in the *Intel Quark* processor, inside the real-time operational system *Viper* from *WindRiver* (acquired by Intel some time ago).
These two processors have an IPC (*Inter-process communication*) channel that works like a FIFO queue, where messages could be exchanged bidirectionally. This pipe channel is used to transfer the image tracking parameters, the telemetry data and the movimentation commands.
The **Process 2** must communicate with the *MultiWii* board too, this is done using a UART (*Universal asynchronous receiver/transmitter*), both the RX and TX channels have FIFO queues internally.

Since we have three components exchanging messages, we should have some time constrain between then, we should follow these steps and work with timeouts read and write operations, avoiding that a process get blocked:

- **Process 2** reads the en_track signal from the RX module and telemetry data from MultiWii board.
- **Process 2** sends these data to **Process 1** using the pipe channel.
- **Process 1** evaluates the received data, do the necessary processing and send new data back to **Process 2**.
- **Process 2** send commands to *MultiWii* board if applicable.



7. References and used Resources

1. http://www.mouser.com/images/microsites/Intel_EDI1ARDUINALK.jpg (Figure 1)
2. http://artofcircuits.com/wp-content/uploads/2014/11/MultiWii-SE-2V5-5.jpg (Figure 2)
3. http://www.mdpi.com/1424-8220/15/12/29785/htm (Figure 3)
4. https://github.com/sol-prog/OpenCV-red-circle-detection
5. https://commons.wikimedia.org/wiki/File:USS_Hunley_%28AS-31%29_top_view_1980.jpeg (Figure 7)
6. https://cse.sc.edu/~yiannisr/774/2014/Lectures/15-Quadrotors.pdf
7. https://github.com/dch33/Quad-Sim (MATLAB Model)