Quadcopter Delivery Verification

Mock Design Review

ECE 445 SP2016

TA: Luke Wendt

Team Members: Sachin Weerasooriya - Electrical Engineering

Raymond Hoagland - Computer Engineering

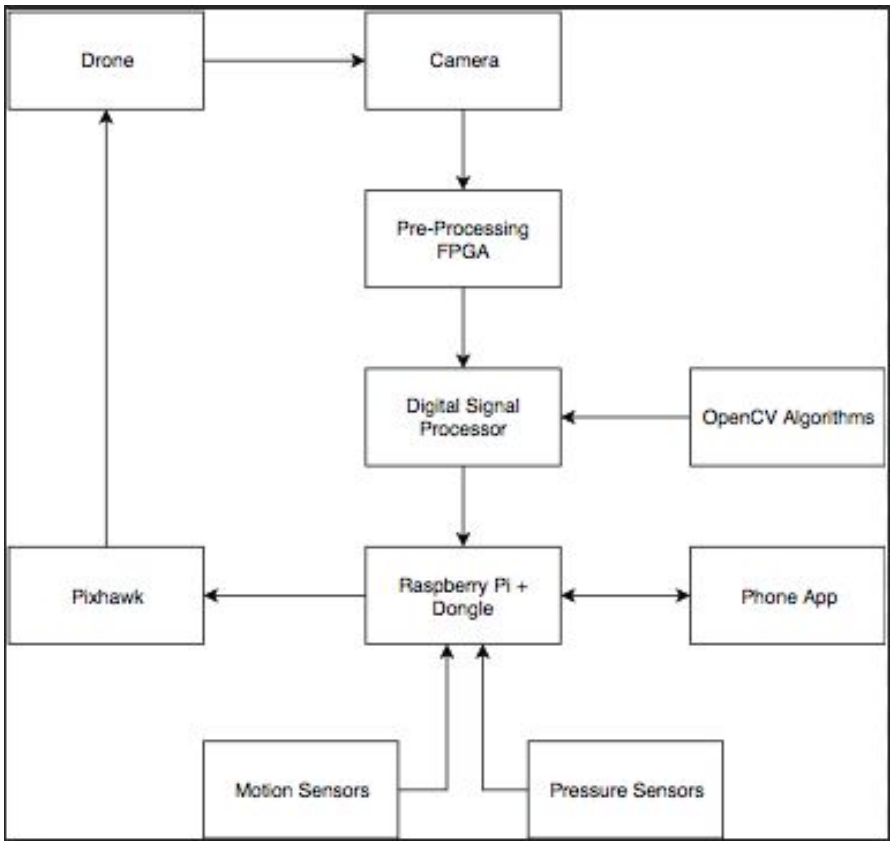
Rahul Joshi - Computer Engineering

February 16th, 2016

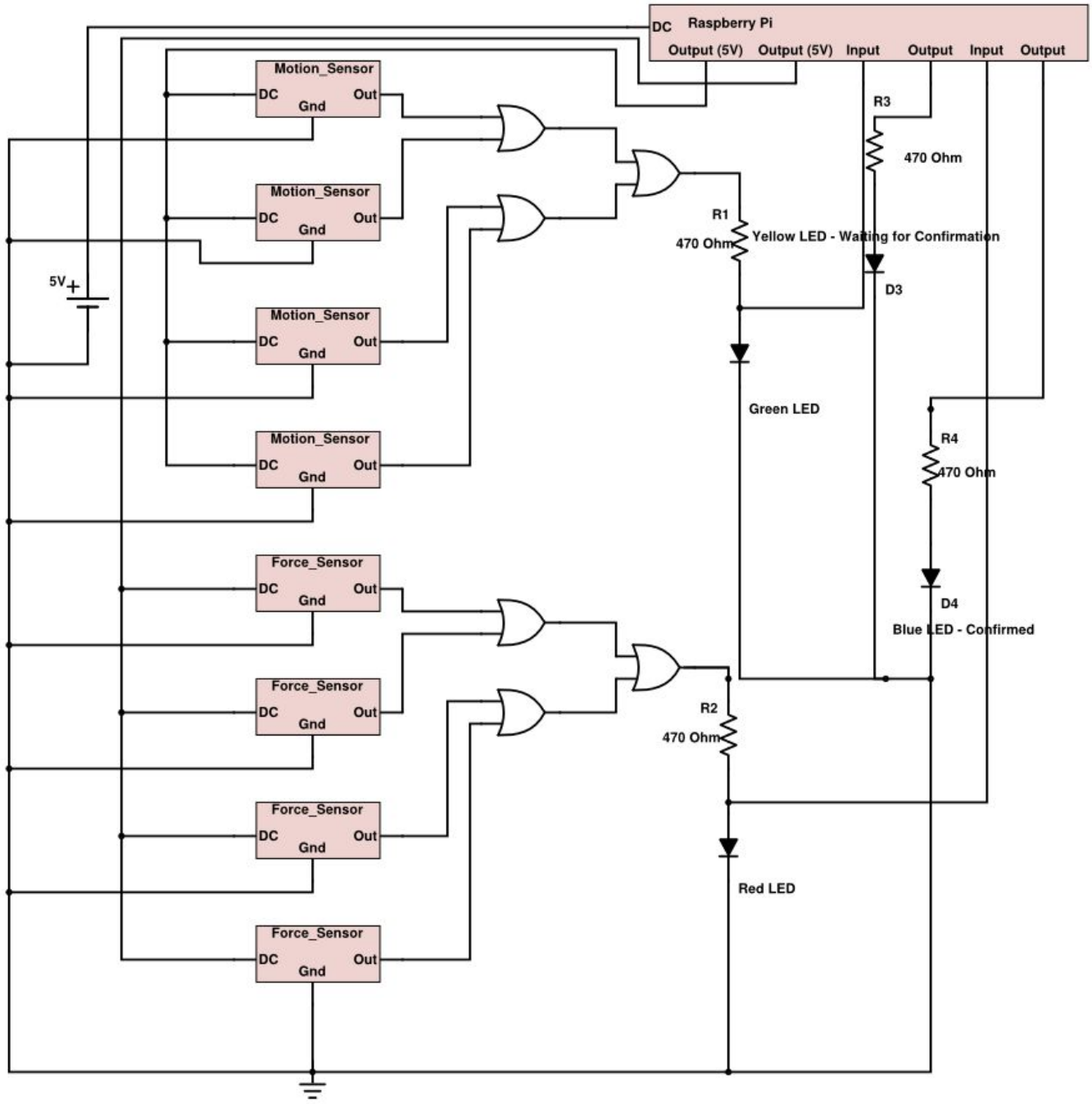
Table of Contents

1 Block Diagram.....2
2 Schematic.....3
3 Block Description.....4
4 Calculation.....5
5 Simulation......6
6 Requirements and Verification.....7
7 Safety.....8
8 References.....9

Block Diagram:



Schematic:


DC Raspberry Pi
Output (5V) Output (5V) Input Output Input Output
Motion_Sensor
DC Out Gnd
Motion_Sensor
DC Out Gnd
Motion_Sensor
DC Out Gnd
Motion_Sensor
DC Out Gnd
5V
Force_Sensor
DC Out Gnd
Force_Sensor
DC Out Gnd
Force_Sensor
DC Out Gnd
Force_Sensor
DC Out Gnd
R3
470 Ohm
R1
470 Ohm
Yellow LED - Waiting for Confirmation
D3
Green LED
R4
470 Ohm
D4
Blue LED - Confirmed
R2
470 Ohm
Red LED

Block Description:

Raspberry Pi + Dongle: Once the drone has reached the designated drop off point, it needs to notify the recipient. The raspberry pi and dongle will send a notification to the recipient's phone app to indicate arrival. The dongle will also receive confirmation that the recipient is ready to pick up the package, indicating that the drone can leave. The Pi will also receive the error signals generated from the DSP. Depending on these error signals, it will generate signals to send to the Pixhawk to order throttle power for each rotor. The Pi will also receive signals from the motion sensors and force sensors to adjust the throttle power as well. If the motion sensors are triggered, the drone will take off. If the force sensors - on the bottom of the drone legs - are triggered, the drone will power down.

Simulation:



Figure 1 Initial Distance from Camera



Figure 2 Second Distance from Camera

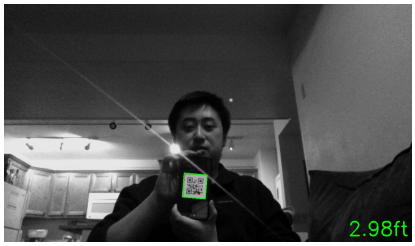


Figure 3 Third Distance from Camera

The above images were taken from three different distances, each progressively farther away. Using the predetermined length of the QR code from the camera for the first image, we were able to determine the focal length and distance from the camera for each of the next images. These images are to help figure out at what height the QR code will still be recognizable. We wish to determine the ratio between the area of the QR-code and the distance from the ground. Using this ratio, we can determine the maximum height at which the drone can scan the QR-code.



Figure 4 Testing QR Code

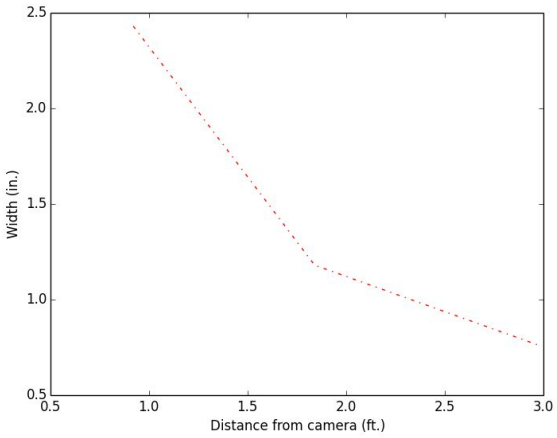


Figure 5 Width vs. Distance

Calculation:

As can be seen in the simulation above, image processing is only feasible when the QR is large enough to be processed by the computer. For this reason, it is important that we calculate the height at which we can fly and still read a QR code.

We plan to establish a landing spot by having a unique pattern on the border of the landing zone along with a QR code in the middle. It is important that we know how high up the drone can be and still detect the border and the QR code. If the drone can't detect the border, it will never be able to properly determine the landing spot. Once the drone finds the border, it will use feedback to move to the middle of said pattern. The drone will then lower to a height it can read the QR code. If it needs to go too low, it will waste precious power but if it stays too high, it risks not being able to read the QR code. For this reason, it is important to calculate the height the drone can fly at and still use image processing to identify objects.

You can then calculate the distance from the camera to the image assuming you know the focal length and width of the QR code in your image.
The formula is:

Distance to image = (knownWidth * focalLength) / (actualWidth)

In the pictures above we took images from 11, 22 and 33 inches away from the screen. Using the above formula, our algorithm calculated 11, 22, and 35 inches respectively so we know it is a relatively accurate measure.

For the farthest away image, we had a percent error of (35-33)/33 = 6.06%

Requirements and Verification:

Requirements	Verification
Raspberry Pi ● Process error signals from DSP to generate signals for each motor to adjust course appropriately ● Process signals from force and motion sensors ○ Hover if detect motion and confirmation hasn't been received ● Also responsible for sending push notification to user app over WiFi and receiving user confirmation ● Send signals to PixHawk Controller based on received signals from DSP	1. Move drone in all four directions 10m using the software on the pi to control the PixHawk Controller 2. Manually trigger GPIO pin input for destination arrival, send text message to verify correct logic triggered. 3. Ensure LED switched on when user confirms delivery.

Safety:

1. Before operating the drone ensure the following checks have been completed.
    a. Drone is started from the off position.
    b. All wires are cleanly soldered and are not loose.
    c. Ensure the drone is programmed to stay within a 5 ft radius geofence and will return automatically if attempting to leave
    d. No personnel are within the geofence and there are no objects impeding the drone's takeoff and touchdown areas.
    e. New code has been peer reviewed and passes automatic testing
    f. Override kill switch is functioning correctly.
    g. Ensure the drone is fully charged.

2. During operation of the drone please be aware of the following safety hazards.
    a. The drone is completely controlled by software as no SIM card will be installed. Thus if the drone is not behaving as it should in any way, send the kill signal.
    b. The drone has a very limited battery so for testing purposes make sure to keep the drone within visual range at all times.
    c. Before approaching the drone make sure it is completely powered off or the verification code has been received. Perceived motion will cause the drone to rapidly elevate which may pose a safety threat to those in the immediate vicinity.

References:

[1] Pixhawk controller diagram
http://copter.ardupilot.com/wiki/introduction/

[2] OpenCV QR-code simulation
http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html
https://ralgozino.wordpress.com/2011/06/13/how-to-create-and-decode-a-qr-code-in-python-using-qrtools/
http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/
http://docs.opencv.org/master/da/d6e/tutorial_py_geometric_transformations.html