Avoiding General Purpose Microcontrollers For Multirotor Autonomous Flight

Innocent Niyibizi
Computer Science Department
Missouri University of Science and Technology
Rolla, MO 65409
Email: iniyibizi@mst.edu

***Abstract*—As the world continues to develop, more and more problems arise that can be solved with the use of advanced software systems. These problems can be things such as navigating an unfamiliar environment or surveying known environments after disaster strikes. To accomplish these goals, months of R&D is poured into specific problems and how to solve them with the current technological advances.**

I. Introduction

This is where the introduction will go for the paper.

1) Brief overview of the problem at hand
2) Introduce the competition of IARC

This opens up the doors to many applications that can be had. A design team on campus, The Multirotor Robot Design Team is one such team and has been tackling this since 2016. They have gone through many microcontrollers and have found the feasibility of each one for the given task.

II. History

This is where the history of each micro controller will be brought up and the team's usage for each one.

1) APM Flight Board, not even close
2) Arduino with no other sensors, disaster
3) PixHawk with some sensors, getting better
    a) Sonar Sensors
4) Pixhawk with sensors but trying to write low level code, Questionable
    a) Optical Flow
    b) Rangefinder
    c) Lidar
5) Pixhawk with sensors and built in functions, beautiful
    a) Optical Flow
    b) Rangefinder

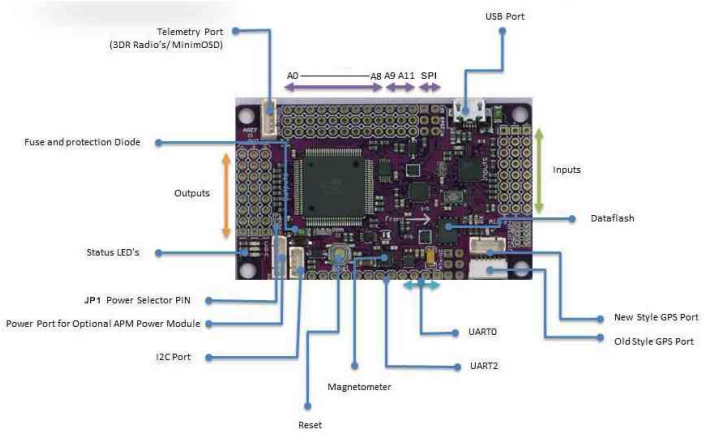

Fig. 1. Layout of APM2.5 Flight Controller

III. The Microcontrollers That Could

A. APM Flight Board: Captain

1) Features
2) Use Cases
3) Practicality of task at hand

B. Arduino: Mega

1) Features
2) Use Cases
3) Practicality of task at hand

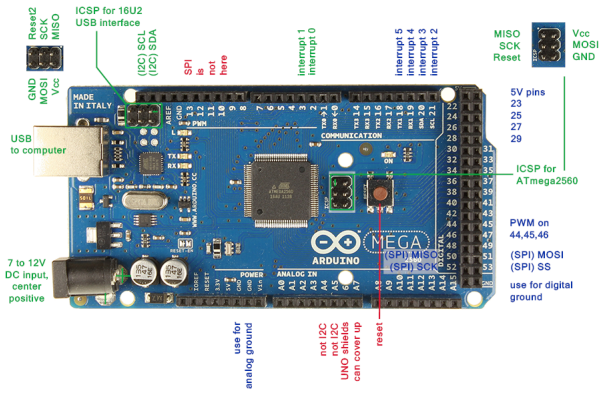

Fig. 2. Layout of Arduino Mega Microcontroller

C. Cortex-M4: PixHawk 2 (The Cube)

This section should include an image of the design

1) Features
2) Use Cases
3) Practicality of task at hand

IV. Conclusion

This section is where the paper will end and everything will make sense once again

References

[1] Bootlin *Linux source code: (v4.18.15) - Bootlin. [Online]. Available: https://elixir.bootlin.com/linux/v2.6.36/source/kernel/sched.c* [Accessed: 07-Dec-2018].

[2] CFS Scheduler *[Online]. Available: https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt* [Accessed: 07-Dec-2018].

[3] L. Jelenković, S. Groš, and D. Jakobović *Testing task schedulers on Linux system.*

[4] J. Morris *"Red Black Trees," Data Structures and Algorithms: Hash Tables. [Online]. Available: https://www.cs.auckland.ac.nz/software/AlgAnim/red_black.html.* [Accessed: 07-Dec-2018]