Dynamic Video Streaming for Nano Quadcopters

Shane Kent
Ryan Weideman
Nicholas Kimball

Computer Engineering
California Polytechnic State University, San Luis Obispo

Senior Project Report
Advisor: Dr. Andrew Danowitz
10 June 2018

ABSTRACT

The objective of this project was to develop a system that streams real-time video from a Crazyflie 2.0 nano quadcopter. We discuss the motivation behind the project including applications to swarm robotics and computer vision research. We highlight the initial research and design goals that guided the development of the system such as hardware selection and system specifications. We detail the software and hardware subsystems that we implemented including the video-streaming board, firmware, and video-streaming user application. We examine the performance of the final system and discuss the limitations imposed by the hardware. We conclude by describing future work that will enhance the capabilities and robustness of the system.

ACKNOWLEDGEMENTS

We would like to acknowledge our industry sponsors Bay Area Circuits, OSH Stencils, and Bitcraze. Without their excellent, professional support, our project would not have been possible. We would also like to thank the Computer Engineering Department for aiding us with extra funding which played a large role in the hardware manufacturing and assembly portion of our project. Finally, we would like to thank Dr. Andrew Danowitz for his support and advice over the course of this project.

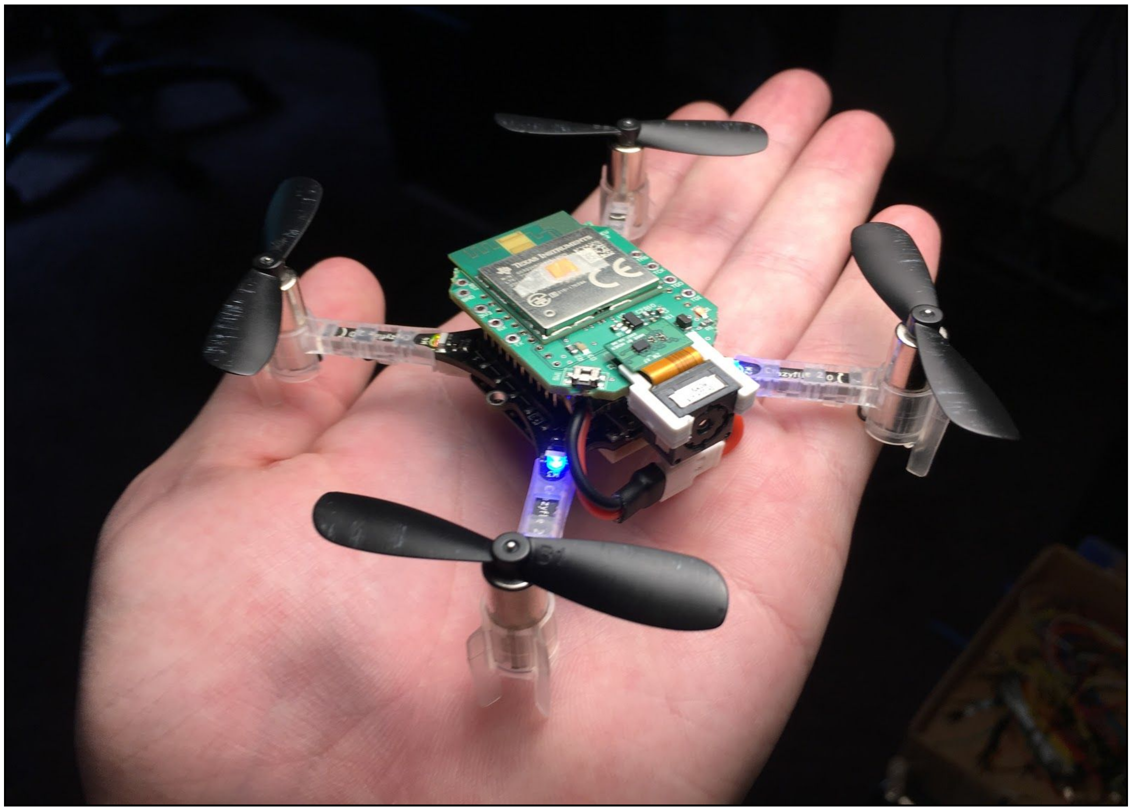


Figure 1: A Crazyflie 2.0 outfitted with custom video streaming hardware.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The overarching goal of this senior project was to have a completed, marketable product with as few loose ends as possible by its conclusion. However, this goal was balanced with the desire to have a senior project which was complex and exciting enough to impress members of the tech-world that were interested in the type of technology being developed. Satisfying these two opposing goals required narrowing the scope of the project such that the work could be finished in the relatively short amount of time of two quarters.

Being a Computer Engineering-based project, we felt it necessary to have a solid mix of both hardware and software components embedded in the project requirements. In addition, we wanted the project to involve topics we were interested in such as robotics and computer vision. Having worked previously on projects with quadcopters, we felt that they provided an excellent platform for robotics research and project development. However, many of the previous projects involved larger quadcopters which suffered from both administrative and technical issues due to the nature of large, flying machines. Wanting to avoid this, we brainstormed for possible projects that could be done with a smaller quadcopter. During this research phase, we came across a video of a TED Talk in which a swarm of hundreds of nano quadcopters autonomously flew in an intricate pattern around the presentation hall[1]. We were inspired by this and decided to pursue a project that would involve nano quadcopter swarms. In addition to the swarm capability, we wanted to include a computer vision aspect to the project. Having experience from a previous project involving a vision system on a quadcopter, we knew that a vision system by itself was a large undertaking. In the theme of narrowing the scope of the project, we decided to instead focus on developing a vision system for a specific nano quadcopter platform - the Bitcraze Crazyflie 2.0[2]. We also discussed, in the event that we finished the vision system early, utilizing our vision system with a positioning system developed by Bitcraze to implement a nano quadcopter swarm for dynamic video streaming. While we did not have enough time to achieve this last step, we feel that the video streaming system we developed achieved both of the original objectives that were previously laid out.

The following sections of this paper delve into the specific aspects of the project. We discuss further background information on the project such as why we decided to focus on the vision system. We detail various phases of the design process and justify our design decisions. We analyze the results of the project and lay out future work that can stem from this project.

2. BACKGROUND

The recent decade has seen an explosion in the popularity of quadcopters driven by their versatility and ease of use. One need only look at the wide variety of industries quadcopters have been adopted into to realize their utility. The quadcopter market can be broken down into several sectors including hobbyist, cinematographic, surveillance, and military. While the cinema, surveillance, and military industries have access to advanced vision systems for their respective specialized needs, the hobbyist market has been stuck with older, low-quality analog video streaming systems. As a result, students and researchers that do not have access to extensive funding are unable to afford the high quality vision systems that are available in the professional market.

The limited access to affordable but high-fidelity vision systems is an issue for students and researchers. Many robotics research problems revolve around making machines autonomous, including quadcopters. A capable vision system is an integral part of an autonomous quadcopter system as it provides the necessary information for path planning, collision detection, object detection, and other vital behaviors.

The focus of this project was to give students and researchers access to a suitable vision system for their quadcopters. The Bitcraze Crazyflie 2.0 (referenced to simply as Crazyflie from this point on) is a popular nano quadcopter which is widely used by hobbyists and researchers because of both its low cost and open-source software. However, there was no vision system for the Crazyflie. Thus, developing a vision system for the Crazyflie platform would allow people interested in robotics but lacking the funding to buy expensive hardware to have a small yet powerful system for use in robotics research.

3. DESIGN SPECIFICATIONS

Table I below provides insight into the initial design requirements that were decided upon before specific hardware was procured and designed.

Table I
Initial design requirements for hardware implementation

Specification	Requirement
Maximum Cross Section of PCB	1.75”
Minimum Hover Time with Stock Battery	4 minutes
WiFi Packet Type	UDP
Minimum Framerate	15 FPS
Minimum Image Size	400x400
Interface to Crazyflie Platform	Crazyflie Breakout Pins
Camera Interface Type	8-Bit Parallel Camera
Camera Image Type	RGB565 or JPEG

The custom hardware design for this project needed to fit within the constrained Crazyflie platform shown in Figure 2 below. The Crazyflie platform is designed with a multi-layer approach where external PCBs known as expansion decks can be attached to the main Crazyflie board via expansion headers. This project was designed to fit into the multi-layer stackup shown in Figure 3, the bottom expansion board was to be left up to the user and the top expansion board would be the custom board designed for this project.

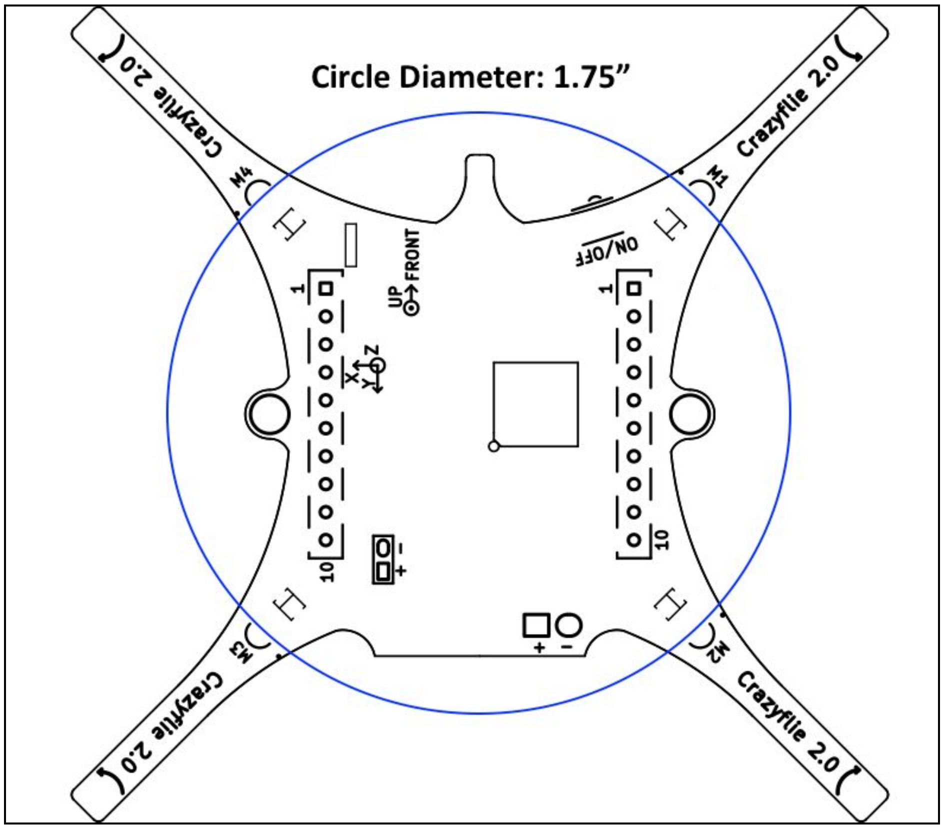


Figure 2: Available real estate for a custom board mounted onto the Crazyflie[4].

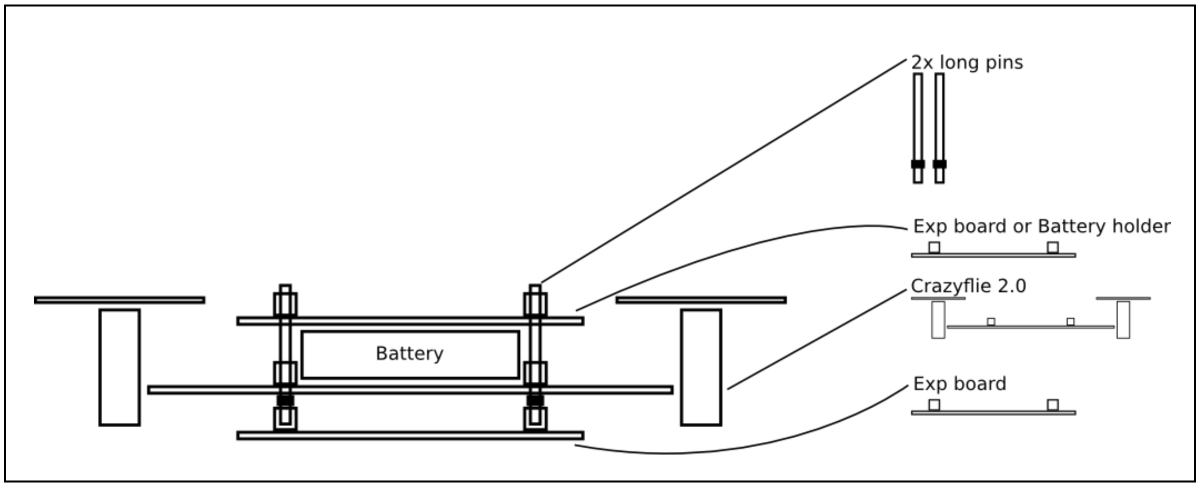


Figure 3: Multi-layer stackup used in the Crazyflie platform[4].

4. APPLICATION OF APPROACH

The design of this project is split into three main categories: hardware, firmware, and software. Below is a discussion of the intricacies and design decisions that were made for each of these categories.

4.1. HARDWARE

The hardware section of the project involved designing a custom board design within the allowed Crazyflie footprint. One of the main design decisions involved choosing appropriate ICs that would allow for an Internet connection over WiFi and a camera interface to get video data from an image sensor. Additional design goals included minimizing the subassembly requirements on the custom PCB to address the power limitations of the on-board battery of the Crazyflie and designing a custom PCB that would be within the standard capabilities of board fabricators like Bay Area Circuits.

4.1.1. Texas Instruments CC3220

There are many different ICs on the market that allow us to meet some of our initial design specifications listed in Table I. However, very few of them allowed for simultaneous integration of both video capturing with a camera interface and video streaming over WiFi.. The Texas Instruments CC3220MODASF SiP module (referred to as CC3220 from here on) allowed for the combination of the networking and camera requirements into one IC without the need for a large amount of support circuitry.

One of the benefits of the CC3220 is that it contains both a WiFi antenna and a parallel camera interface internally such that no support hardware is needed between the camera and the module itself. This was extremely useful because it allowed for a direct interface between gathering data and sending it out over WiFi. There was no need for any support hardware to do this (external memory, WiFi antenna, etc.).

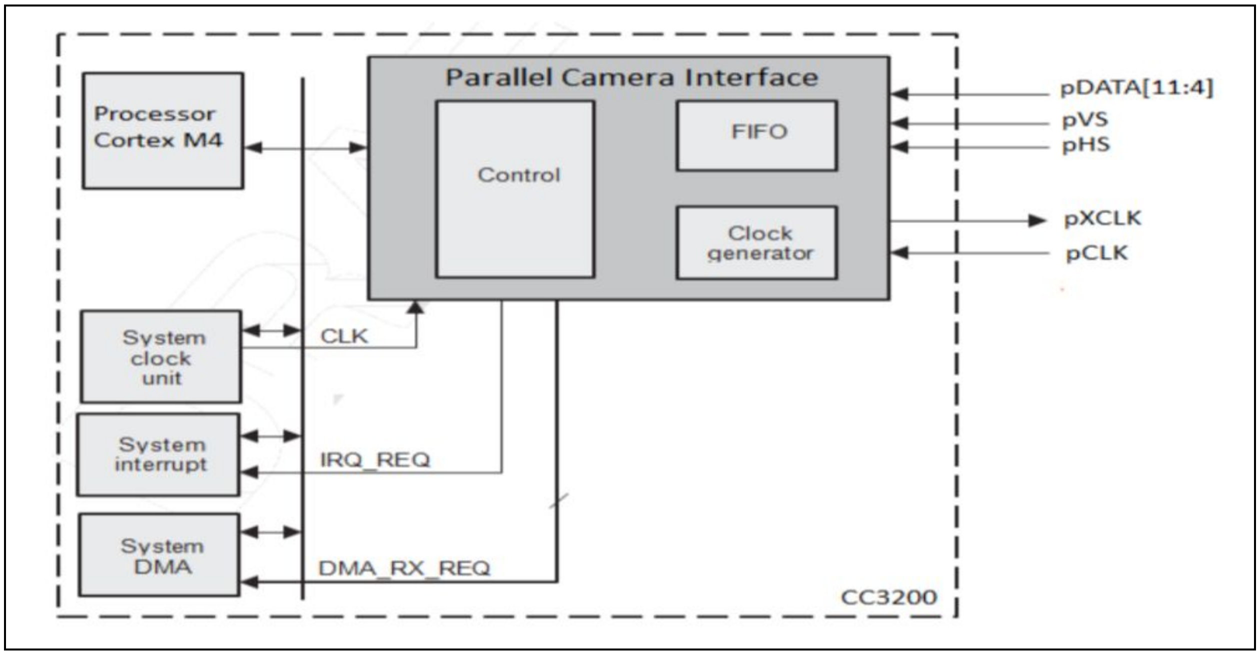


Figure 4: Diagram of the parallel camera interface that is implemented in the CC3220[3].

Another benefit of the CC3220 was its compact design. Because this project required an extremely compact footprint that would be able to fit onto the Crazyflie platform, all of the support hardware including the ICs and the camera module itself needed to be chosen with size as a major consideration. As can be seen in Figure 5 below, the overall package size of the CC3220 module is compact and shaped in a way where it can fit perfectly in-between the mating header connectors of the Crazyflie platform.

Finally, the CC3220MODASF version of the CC3220 family line has the benefit of being a SiP package so that it has all of the required support hardware for the IC in the package itself. This includes the addition of an on-board WiFi antenna which can be extremely difficult to design and implement in a custom design. By using the CC3220MODASF, all of the WiFi functionality of the IC was essentially guaranteed because a custom WiFi antenna was not required.

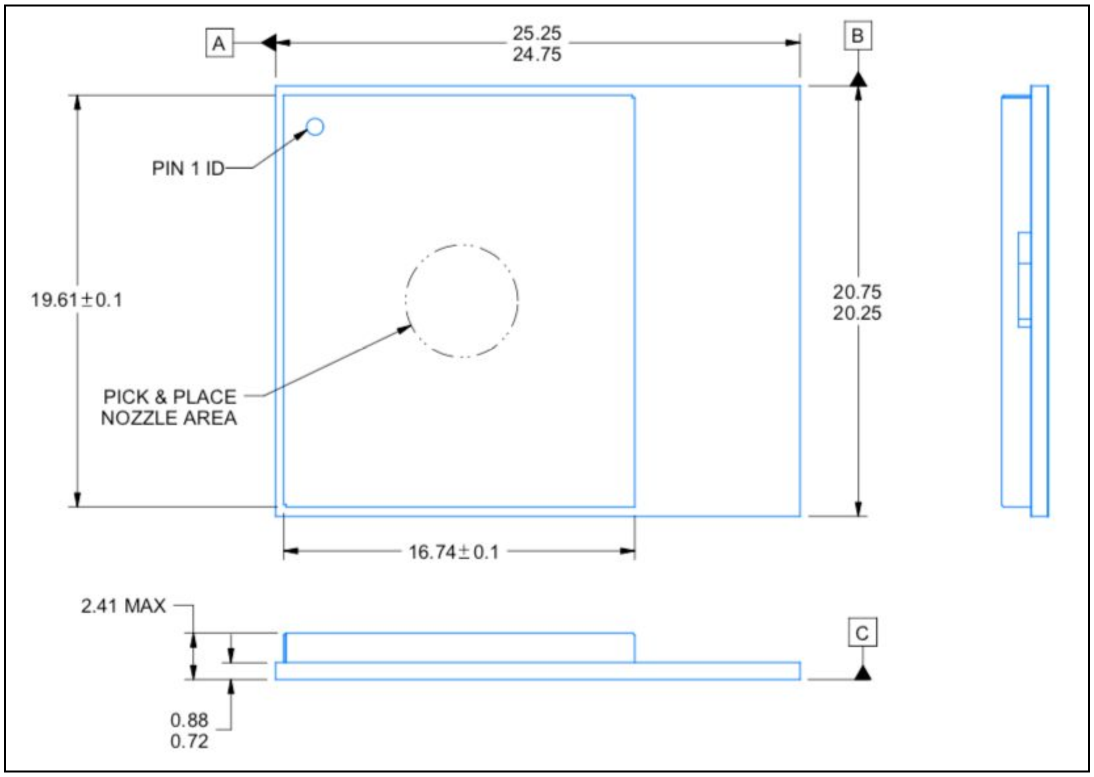


Figure 5: Physical package size of the CC3220 module[3].

A difficulty with the CC3220 module is that it utilizes SMT and as such can be extremely difficult to assemble and debug connections without the proper tools. The assembly portion of this project regarding the CC3220 is discussed in Manufacturability and Assembly section below.

4.1.2. Micron MT9D111 Camera

As camera technology continues to improve, many of the companies that design cameras are only willing to provide documentation for their modules to customers that are planning to procure thousands of units per year. As such, the camera used for this project had to be easily accessible with proper documentation available for the average user. With those requirements for choice of camera as well as the requirements listed in Table I, the Micron MT9D111 camera module (referred to as MT9D111 from here on) was used for this project.

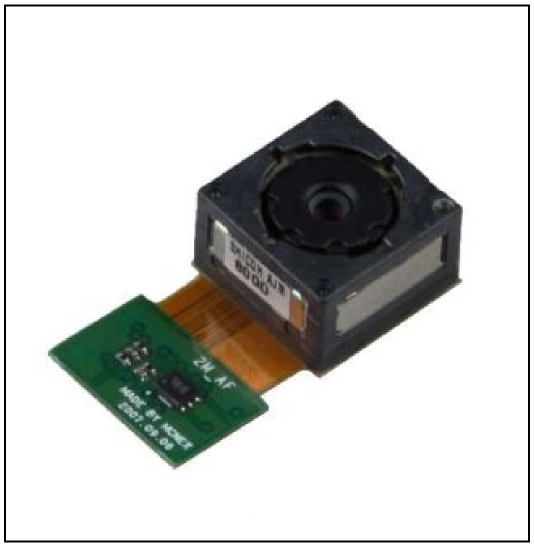


Figure 6: Micron MT9D111 camera module used for this project[5].

The MT9D111 has the benefit of requiring only external voltage regulators and an I2C connection to the on-board MCU. The MT9D111 allows for either a raw RGB565 or JPEG image capture. The electrical requirements for the MT9D111 are shown in Table II below.

Table II
Summary of the electrical requirements for the MT9D111[6]

Specification	Requirement
Maximum Frame Rate	30 FPS in Preview Mode
Master Clock	6 MHz to 80 MHz
Voltage Rails	1.8V, 2.8V, 3.3V
Average Power Consumption	223mW at 30 FPS
Image Output Format	RGB565 or JPEG
Maximum Resolution	1600x1200

The MT9D111 module was discontinued in 2004 and is no longer manufactured. As a result, there is very little support for the image sensor itself other than the original datasheet for the module. This led to some difficulty in determining the proper registers and values to write to the registers to obtain the desired performance.

4.1.3. Power Considerations

The graph shown in Figure 7 below provides some meaningful information that can be used to calculate an expected flight time for Crazyflie when the stock battery is used and the Crazyflie is hovering.

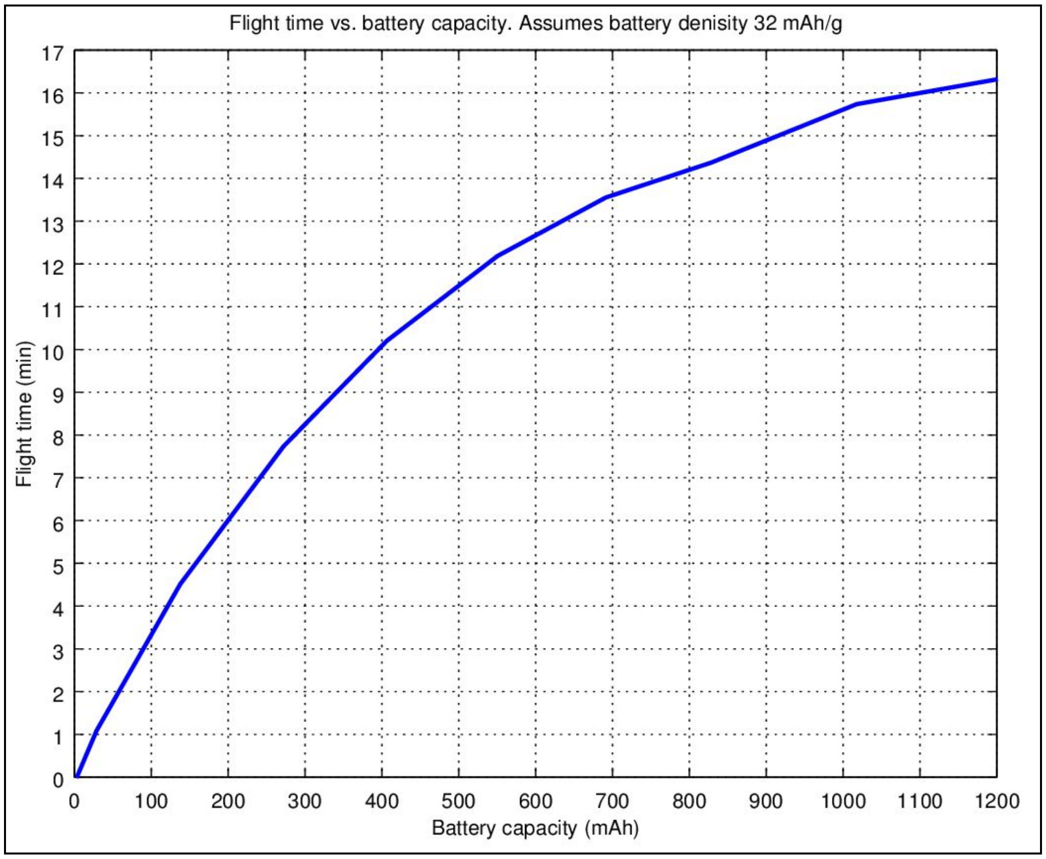


Figure 7: Flight time compared to battery capacity used to estimate expected flight time[7].

The stock battery that comes with the Crazyflie platform is a 3.7V, 240mAh LiPo battery. Using the graph below and the reported default hover time of 7 minutes, the current draw for the default system setup can be calculated as follows:

$$C_{battery}[mAh] = I_{draw}[mA] \ * \ \frac{t_{flight}[ms]}{3{,}600{,}000} ,$$

$$I_{draw} = (240 \ mAh) * \frac{3{,}600{,}000 \ ms/h}{420{,}000 \ ms} = 2057 \ mA$$

The maximum current draw allowed to achieve the minimum 4 minute hover time as described in Table I is calculated as follows:

$$I_{draw} = (240 \ mAh) * \frac{3{,}600{,}000 \ ms/h}{240{,}000 \ ms} = 3600 \ mA$$

This provides an overhead of approximately 1.54A that the custom PCB can draw from the battery and still allow for a theoretical 4 minute hover time.

A test setup was created to measure the actual current draw of the custom PCB for this project. The setup was the designed so that the custom PCB is powered via an external 3.7V power supply to mimic the normal operating voltage of the LiPo battery on the Crazyflie. While the custom board is in setup, standby, and fault states, the custom PCB was measured to draw approximately 85mA. While streaming image frames at approximately 15 FPS at a frame size of 400x300, the current draw remained at approximately constant draw of 145mA. Assuming an extreme case where the custom board draws a constant 175mA, the hover time is calculated as follows:

$$t_{flight}[ms] = 3,600,000 * \frac{C_{battery}[mAh]}{I_{draw}[mA]}$$

$$t_{flight}[ms] = 3,600,000 * \frac{240\ mAh}{2057\ mA + 175\ mA} = 387,096\ ms$$

$$= 6.45\ minutes$$

Thus, calculating the expected hover time while only considering current draw from the custom board yields a decrease in the original hover time of 7.86%. Of course, the hover time of will also be affected by the weight of the custom board, but its effect on the hover time is negligible compared to the overall current draw of the custom board.

4.1.4. Board Design

The custom PCB for this project was designed in Altium Designer 17. Two different revisions were designed and sent out for fabrication. The initial design had a mistake in the connector footprint for the camera module such that the camera connector could not be attached properly. It also had very limited test points for power and programming connections. The driving factor for adding in additional programming connections was because the CC3220 must be flashed once initially when the chip is first used . In order to flash the IC, the CC3220 must be connected to a host computer over a UART connection. The second revision of the board included test point through holes, the appropriate camera connector footprint, and an additional LED on a GPIO pin used as an indicator for the video stream frame rates.

The 3D functionality built into Altium Designer allowed for a clear view of how the board would fit onto the Crazyflie platform. Unfortunately, the lack of accurate 3D models for the Crazyflie meant that modelling an assembly of the Crazyflie and our custom board would not easily be possible. Regardless, the dimensions of the custom PCB needed to be verified on a Crazyflie. In order to do this, the model shown in Figure 8 was 3D printed and the resulting print was mounted on a physical Crazyflie to verify that there were no collisions between the Crazyflie's propellers and any portion of the custom PCB.

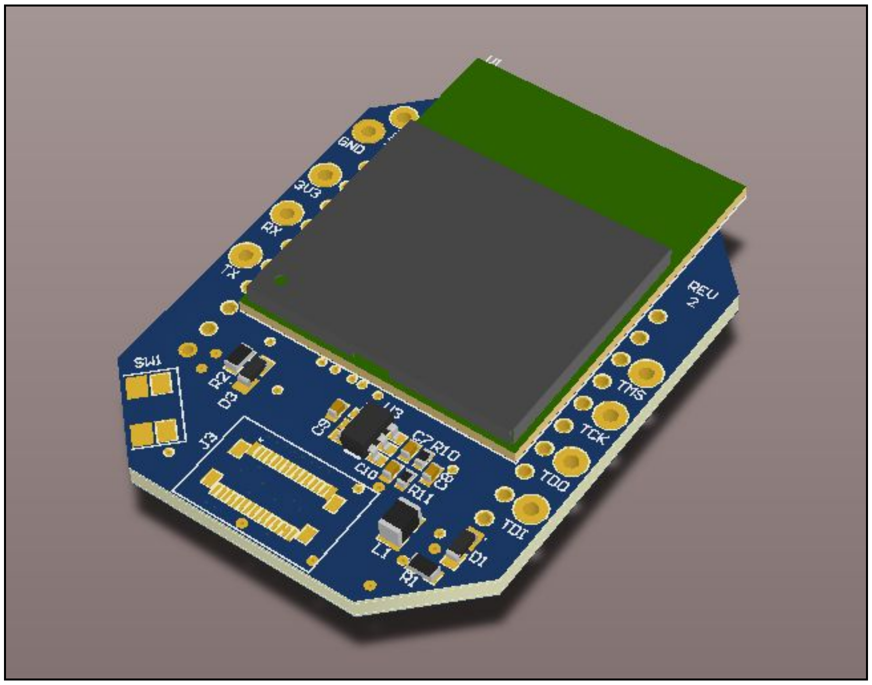


Figure 8: 3D Model generated in Altium Designer showing the custom PCB for this project.

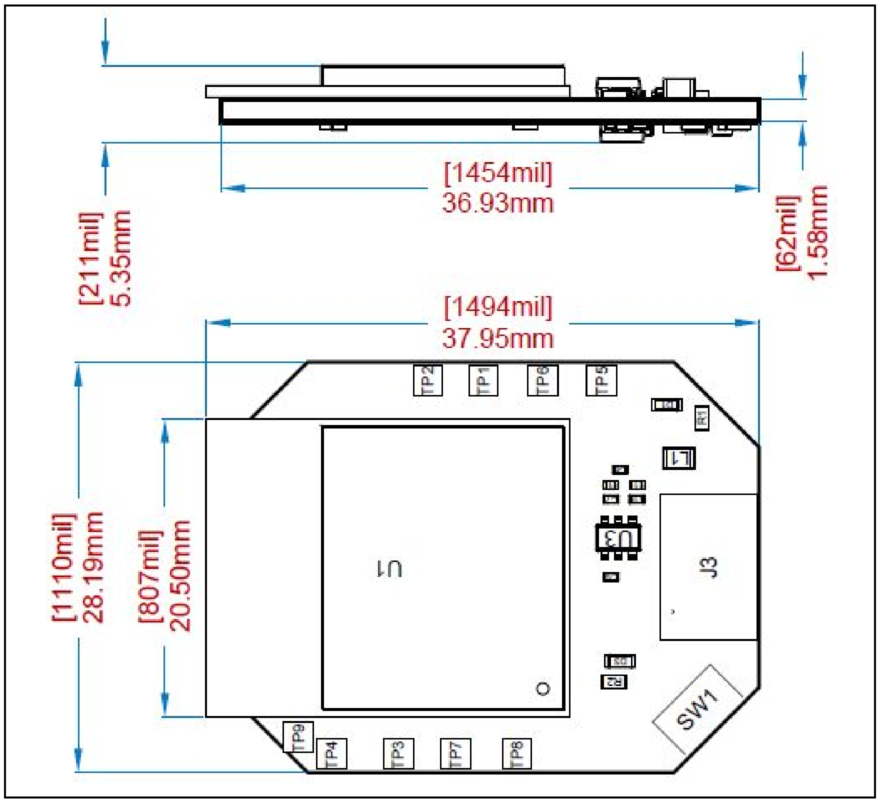
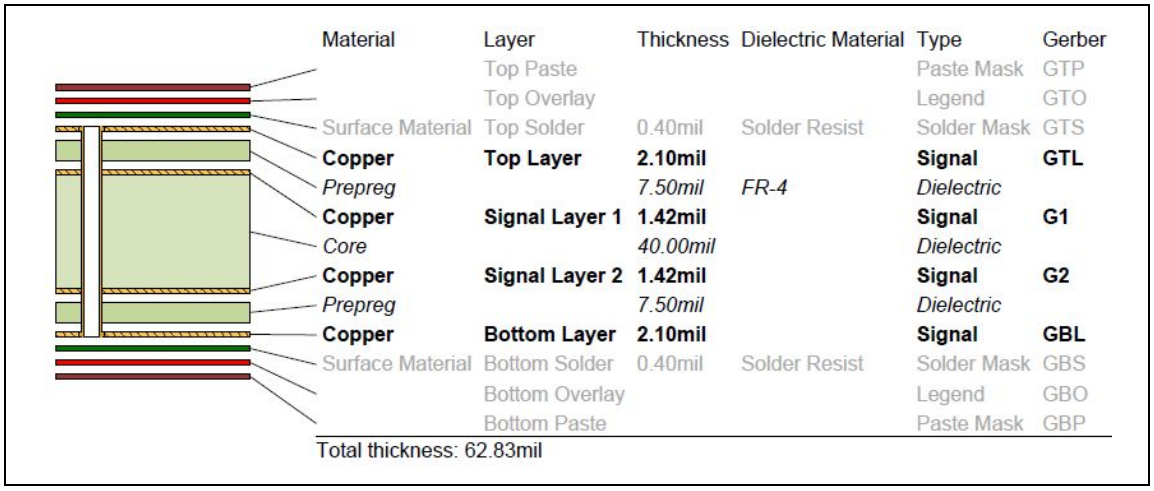


Figure 9: Overall package size of the custom PCB designed in Altium Designer.

As discussed in the Texas Instruments CC3220 section above, the CC3220 is in a compact SMT package with a pad-to-pad pitch that does not easily allow for traces to be routed between two adjacent pads. Because of this a four layer board stackup (shown in Figure 10) was used so that clocks, data lines, and power planes could be separated cleanly and thus avoid crosstalk and other concerns that would otherwise be an issue.



Material	Layer	Thickness	Dielectric Material	Type	Gerber
	Top Paste			Paste Mask	GTP
	Top Overlay			Legend	GTO
Surface Material	Top Solder	0.40mil	Solder Resist	Solder Mask	GTS
Copper	**Top Layer**	**2.10mil**		**Signal**	**GTL**
Prepreg		*7.50mil*	*FR-4*	*Dielectric*	
Copper	**Signal Layer 1**	**1.42mil**		**Signal**	**G1**
Core		*40.00mil*		*Dielectric*	
Copper	**Signal Layer 2**	**1.42mil**		**Signal**	**G2**
Prepreg		*7.50mil*		*Dielectric*	
Copper	**Bottom Layer**	**2.10mil**		**Signal**	**GBL**
Surface Material	Bottom Solder	0.40mil	Solder Resist	Solder Mask	GBS
	Bottom Overlay			Legend	GBO
	Bottom Paste			Paste Mask	GBP

Total thickness: 62.83mil

Figure 10: Layer stackup used for the custom PCB.

Although the stackup shown above was sent as a design requirement to our board fabricator, the actual prepreg and core may vary slightly in their overall thickness and dielectric constant. For this project and application, the variation is not of much concern as impedance control was required by either the CC3220 or the MT9D111.

4.1.5. Manufacturability and Assembly

The manufacturing capabilities for having a board fabricated are decided upon by the manufacturing house used. Bay Area Circuits, one of our sponsors, provides a clear list of their capabilities for their standard and advanced boards on their website[8]. The custom PCB for this project was designed to fit with BACs standard capabilities when possible in order to improve the lead-time for fabrication and decrease the overall cost for having the boards fabricated. However, because the design for this project required a tight footprint, in some cases the actual design required Bay Area Circuit's advanced capabilities. Table III below shows the final design values that were used in the custom PCB for this project. As with many fabricators, the actual design used to fabricate the boards may have been slightly changed by BACs engineers in order to better fit their capabilities. For example, the minimum copper to copper distance in the custom design is 4.93 mil, so in order to better fit their capabilities, BAC retains the right to change that to 6 mil when possible.

Table III
Design choices used for PCB fabrication

Specification	Bay Area Circuits Capabilities	Actual Design
Min. Copper to Board Edge	5 mil	5.0 mil (0.13 mm)
Min. Trace to Plated Hole	5 mil	8.89 mil (0.226 mm)
Min. Copper to Copper	2 mil	4.93 mil (0.125 mm)
Min. Copper Ring	1 mil	3.95 mil (0.100 mm)
Min. Track Width	2 mil	5.0 mil (0.130 mm)

Once the custom PCBs were fabricated, they were assembled in-house using a stencil from one of our sponsors, OSH Park, in order to apply SAC305 solder paste. Once the solder paste was applied, each of the components were placed in their appropriate locations with the help of a microscope. The boards were then reflowed following a standard SAC305 reflow profile shown in Figure 11. Special care was taken when reflowing the custom boards in order to protect the components from heat shock.

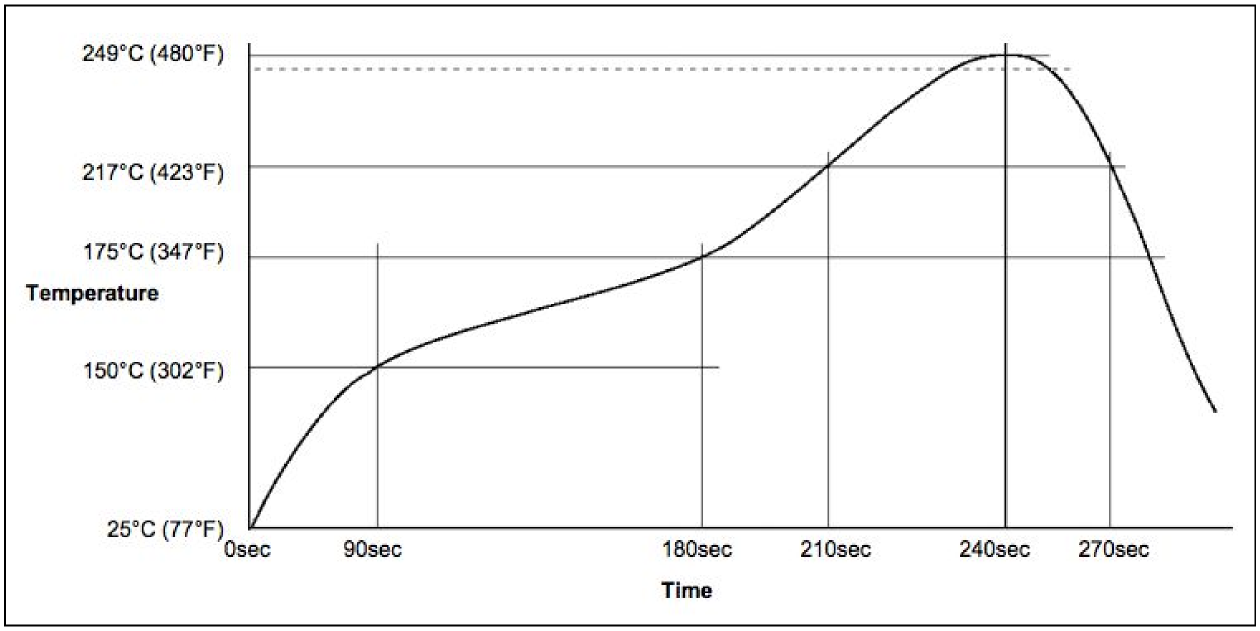


Figure 11: Recommended reflow profile for SAC305 solder paste made by Chip Quik[9].

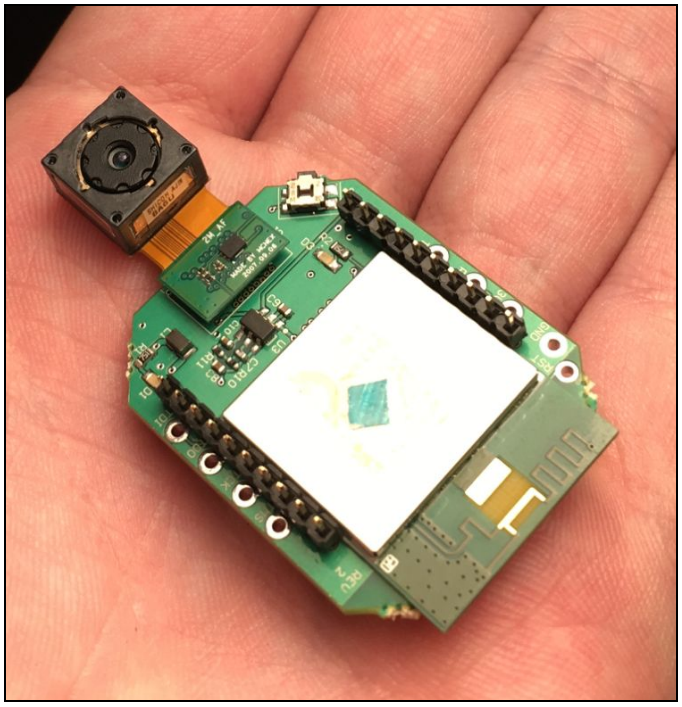


Figure 12: Final custom PCB design.

4.1.6. Mechanical Interface

To mount the camera module to the custom PCB, a small, rigid connector was designed in Autodesk Inventor and 3D printed with PLA plastic using a Monoprice 3D printer. A CAD model of the connector can be see in Figure 13 below. The connector was designed such that the camera is held stable via a clean press fit . It mounts to the PCB such that it does not interfere with any of the SMD components on the board.

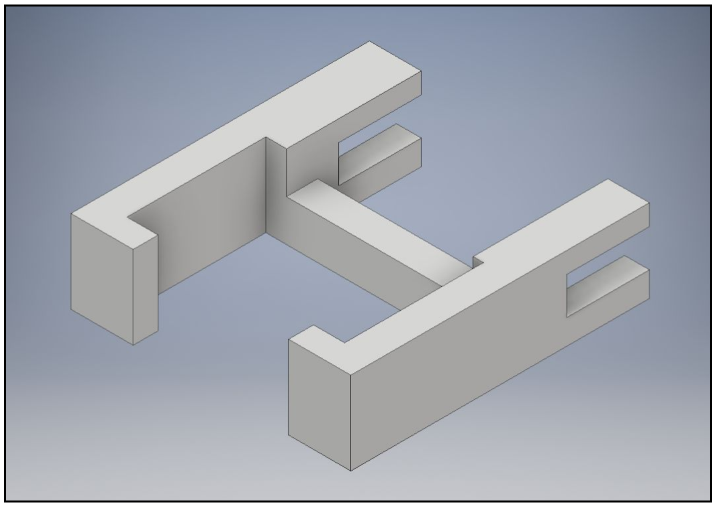


Figure 13: Screenshot of CAD model for the camera mount.

4.2. FIRMWARE

The firmware portion of the project involved designing the code to make the CC3220 initialize the camera to appropriate settings, grab data from the camera, and send the data to a host computer over WiFi. Writing the firmware to interface with the camera proved to be very time consuming due to a lack of helpful documentation and relevant examples. In addition, the camera initialization code often failed during the initial development due to electrical problems in the development board being used. The development of the firmware was mostly done on the CC3220 LaunchPad shown in Figure 14 below, and then was slightly modified to fit the final board.

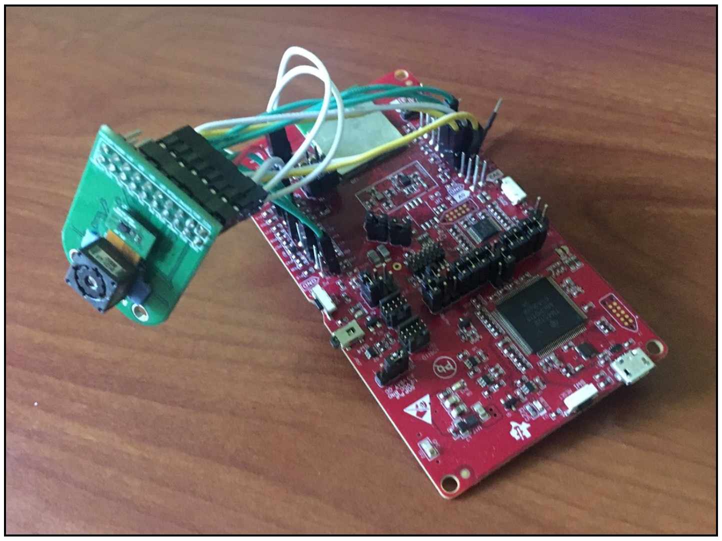


Figure 14: CC3220 LaunchPad and MT9D111 development boards for firmware development.

4.2.1. Camera Firmware

The camera firmware included the code that initially programmed the camera with the appropriate register values as well as the code that read data from the camera. The camera initialization code runs only once and occurs at boot up. The code writes to a series of registers in the camera over an I2C interface between the CC3220 and the camera module. The register writes initialize the camera with the appropriate resolution, frame rate, and image capture mode. During development with the LaunchPad, the initialization of the camera would often fail and was a source of frustration throughout a majority of the project. The problem was likely due to the large impedance and noise present on the testing setup as the final hardware did not have this issue. Regardless, future versions of the code should include logic to deal with the case that the slave I2C device faults and locks the bus up.

4.2.1.1. Parallel Camera Interface

The parallel camera interface on the CC3220 was one of the main reasons the chip was chosen for this project. Camera interfaces such as CSI are now almost exclusively used in consumer electronics due to the higher data rates and smaller footprint possible. However, it is very difficult to find a vendor willing to sell small volumes of modern image sensors that support CSI. As mentioned previously, the only easily available cameras were designed over a decade ago and use the parallel camera interface. This necessitated the use of a chip that supported the parallel camera interface.

The parallel camera interface requires 12 signals. These consist of a master clock (XCLK), a pixel clock (PCLK), a frame valid signal (FV), a line valid signal (LV), and 8 data signals (D0-7). The master clock is generated in the CC3220 and is sent to the MT9D111's PLL where its frequency is multiplied and is used to clock the internal systems of the camera. The pixel clock is generated by the camera and is used to latch the 8 data bits into the CC3220. The frame and line valid signals are generated by the camera and indicate to the CC3220 when the data on the interface is valid and when a single frame is done being transmitted.

The CC3220 has the ability to output a master clock up to 120 MHz. For this project, the master clock frequency was 10 MHz and the camera PLL was programmed to multiply the input frequency by ~1.94 times. Therefore, the internal clock of the camera was around 19.4 MHz. The internal PLL in the camera allows a reduction in EMI as the clock signal generated in the CC3220 does not need to be as high frequency which can help to reduce crosstalk on data lines. The camera documentation states that the camera can run at an internal clock rate as high as 80 MHz and that 36 MHz is the recommended minimum for suggested applications. Unfortunately, as described in the following section, clocking the camera at higher rates led to problems with the CC3220 not being able to handle the data throughput.

4.2.1.2. Pixel Clock Limitations

As mentioned previously, the pixel clock is a signal generated by the camera which latches data into the CC3220 when the line and frame valid signals are also asserted. The technical reference manual for the CC3220 states that the recommended maximum input pixel clock is 1 MHz. However, the datasheet ambiguously claims that the maximum pixel clock is 2 MHz[10]. To further complicate the issue, the one and only TI example application for the parallel camera interface suggested that the pixel clock could handle even higher frequencies. The lack of a concrete specification for the pixel clock was a source of frustration throughout the entire project.

With much troubleshooting and experimentation, the maximum pixel clock was found to be 1.5 MHz. Anything higher than this frequency caused the CC3220 to read in the wrong colors, with higher frequencies causing more and more corruption. This corruption is demonstrated in the pictures below:

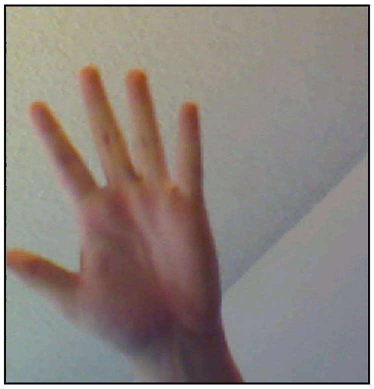


Figure 15
No corruption at 1.5 MHz

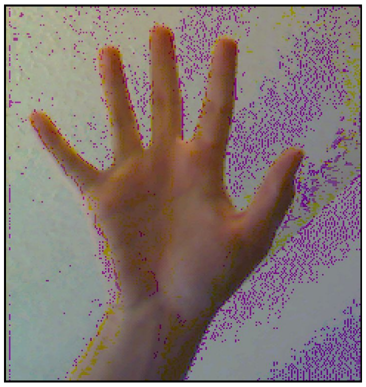


Figure 16
Visible corruption at 2 MHz

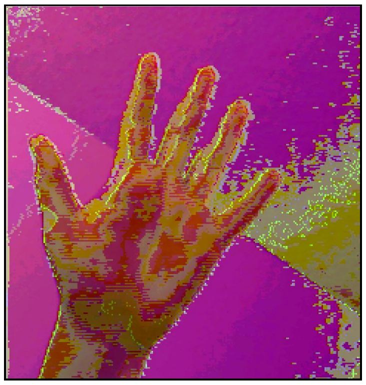


Figure 17
Total corruption at 6 MHz

The limitation on the pixel clock severely limits the amount of data the CC3220 can read over the parallel camera interface. With an 8-bit data bus and a 1.5 MHz pixel clock, the maximum throughput of the CC3220's parallel camera interface is 1.5 MBps. For comparison, the STM32F407 MCU has a maximum pixel clock of 54 MHz which allows up to 54 MBps of throughput. Luckily, the MT9D111 had an internal JPEG compression engine which allowed higher resolutions than would have been possible using raw images. This is discussed in more detail in the following section.

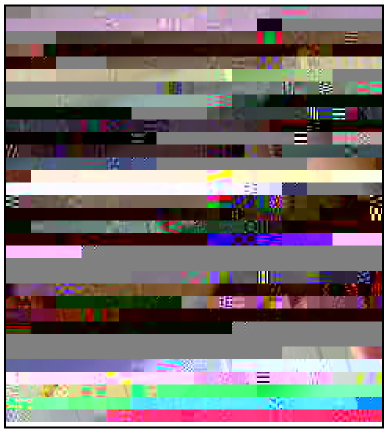


Figure 18: Corrupted JPEG image due to pixel clock being 2 MHz

The above figure shows the effect of the pixel clock corruption on JPEG images. In the raw images, the corruption changes color values but the objects in the image are still discernible. However, when corruption occurs in a compressed image such as JPEG, the decompression algorithm will fail to properly decompress the image, resulting in complete image corruption as seen in the figure above.

Overall, the limitation on the data rate as a result of the pixel clock was one of the largest constraints in this project. The maximum pixel clock for any future designs will be a main point of consideration when making hardware selections.

4.2.1.3. JPEG Compression

Due to the pixel clock limitations discussed and the available WiFi bandwidth on the CC3220, it was necessary to compress each of the image frames. This reduces the overall amount of data output from the camera which enables the CC3220 to handle higher frame rates and resolutions. The MT9D111 camera module features an integrated JPEG encoder that allows frames to be compressed with JPEG within the camera module itself. Enabling this feature partially alleviates the limitation imposed by the CC3220’s maximum pixel clock as the compression enables a reduction in frame size by up to ten times. When compressing frames, the camera’s JPEG engine uses the standard quantization tables and Huffman coding tables. The quality of the JPEG images (Q-factor) can be configured in the camera’s registers where more compression leads to smaller but lower quality individual frames. Compressed video frames are output from the camera module and are read in via the CC3220’s parallel camera interface. In order for the JPEG decompression algorithm to work, each JPEG image must contain a header that defines its resolution, quantization and Huffman coding tables, and other metadata about the image. In this system, the JPEG header is not output by the camera on every frame. Instead the header is prepended to the compressed frame before decompression in the user application, further reducing the amount of data necessary to be output and sent over the network. In this project, the ability to compress frames was absolutely essential to achieving a video stream of reasonable quality, frame rate, and resolution.

4.2.2. Simplelink Wireless Networking

The CC3220 features WiFi capabilities through the Simplelink device internal to the chip. TI provides firmware to interface the device and to provide a full networking stack. This makes it possible to write high level firmware to connect to a wireless LAN, and quickly begin writing networking code. In this application, compressed frames are broken into UDP datagrams that are 409 bytes long. These packets are sent to a firmware hardcoded WLAN address and specific host IP address.

4.2.3. Networking Protocol

A simple networking protocol was established on top the UDP protocol in order to standardize information essential to the transmission of video frames. The custom header (shown in Figure 19) in the data section of the UDP datagram contains information that the receiving end needs in order to ensure the entire frame has been sent and can be decompressed, along with an identifier specifying which Crazyflie the frame is coming from. Thus, each datagram is 409 bytes with 400 bytes of data and 9 bytes of header.

Crazyflie ID - 8 bit	Frame Length - 32 bit	Packet ID - 16 bit	Packet Length - 16 bit

Figure 19: Structure of data packets sent over UDP for each video stream to host computer.

The Crazyflie ID is a one byte value at the head of the packet which indicates which Crazyflie the video feed is arriving from. This ID is hardcoded into each Crazyflie's firmware. This one byte value allows for 256 unique Crazyflies to stream video to the same host computer at once. However, this value can easily grow to accommodate more Crazyflies in the future if more than 256 unique Crazyflies are to stream at once.

The Frame Length field is the total length of the compressed frame in bytes, excluding the JPEG header. This value is used to determine how many UDP packets of frame data the user application should expect.

The Packet ID field identifies the number of the packet within a frame. This value increases sequentially with each packet of frame data sent, starting with zero at the first packet of the frame. This value is used by the receiving application to determine if it has properly received all the compressed frame data.

The Packet Length field is the length of the current UDP packet. This value is used as a redundancy check for the user application.

4.3. SOFTWARE

In order to receive the video streamed by each Crazyflie, a custom application was developed that allows users to watch the video feed from any particular Crazyflie. The software also allows the user to cycle through all of the currently streaming Crayzflies, and display relevant system statistics, such as the current number of Crazyflies and the amount of data being received by the application. This application was developed in Java for its speed, standardized libraries, and ease of multi-platform distribution.

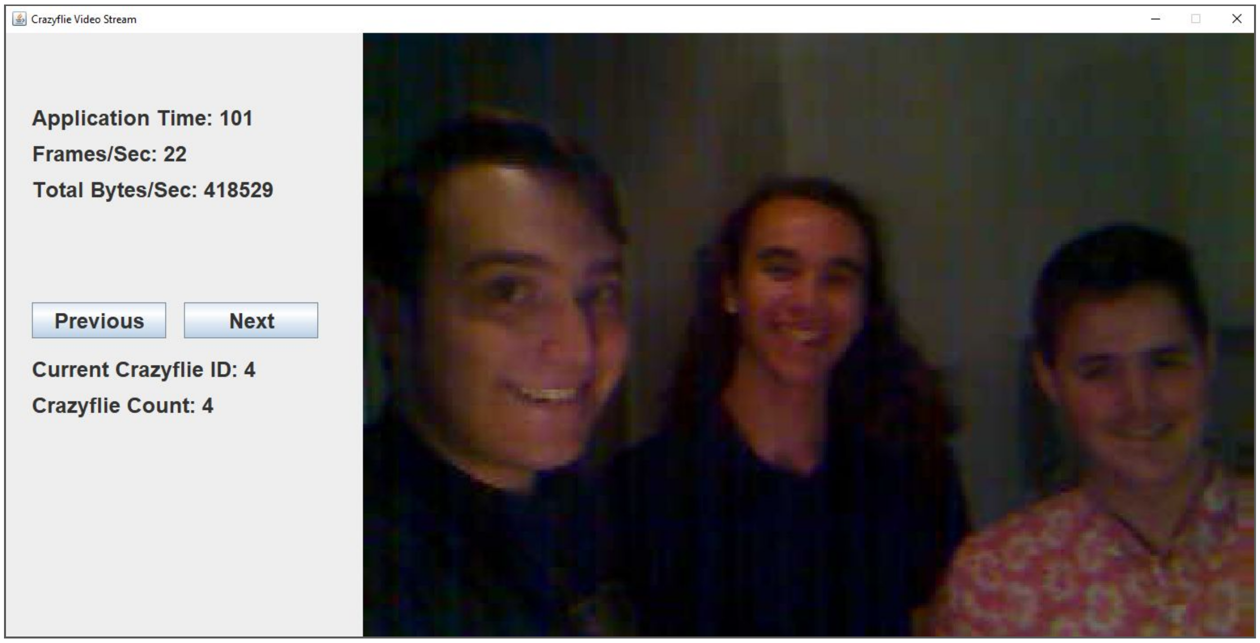


Figure 20: Screenshot of final host application software developed for this project.

4.3.1. Networking

The application opens a socket on Port 5006 and continually polls for arriving UDP datagrams containing frame data sent from the Crazyflies. The packet header is then processed to retrieve the information outlined in Section 4.2.3. The Crazyflie ID is used to keep track of the Crazyflies that are currently streaming to the application; if a Crazyflie has not streamed data to the application for seven seconds, the application considers the Crazyflie to be disconnected. The ID is also used to determine which video stream should be displayed. The video stream can be cycled between multiple Crazyflies by pressing the "Previous" and "Next" buttons.

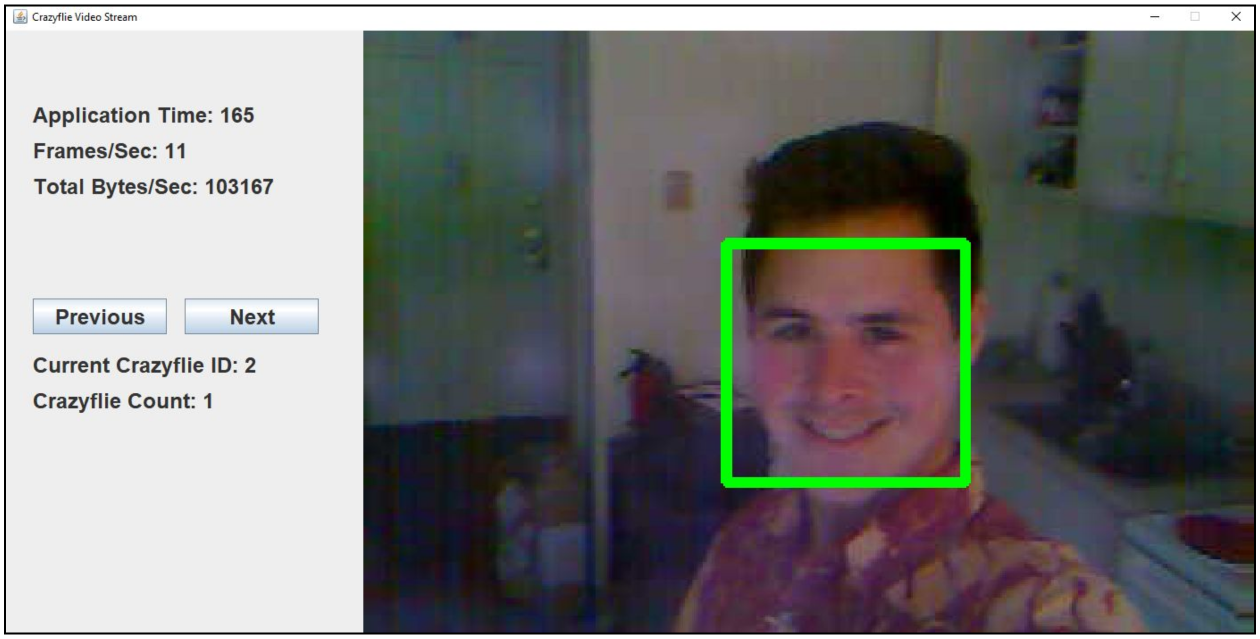
The other fields in the packet header are used to ensure that the entire frame has been successfully received and buffered. The application does this by keeping track of what packet ID is expected next, and buffering the packet data if the packet contained the correct ID. If the packet ID of a received packet is not the expected ID, then the packet data is not buffered and the entire frame is considered lost. While this may seem wasteful, it ensures speed (the application does not need to communicate to the Crazyflie that it has lost a packet) and that all compressed frame data is received correctly. Without this guarantee, JPEG decompression of the buffered frame data will fail.

4.3.1.1. JPEG Decompression

Before the compressed JPEG data received from the network can be decompressed, a JPEG header with the proper fields must be prepended to the data. This header is generated from TI's CC3200 MT9D111 example firmware project and is hardcoded into the application. Once the header has been prepended to the data, the data becomes a valid JPEG file that can then be decompressed. OpenCV, an open source computer vision library, is used to decompress the JPEG frame into a standard RGB bitmap at the resolution specified in the JPEG header.

4.3.1.2. Image Processing and Facial Recognition

To prove the quality of the images transmitted from the Crazyflie, the application can perform post-compression image processing and facial recognition using OpenCV. This demo proves that images received are quality enough for some applications that researchers or hobbyists may choose to use the Crazyflie for. Figure 21 illustrates the Local Binary Patterns visual descriptors in OpenCV for facial recognition.



Application Time: 165
Frames/Sec: 11
Total Bytes/Sec: 103167

Current Crazyflie ID: 2
Crazyflie Count: 1

Figure 21: Screenshot of facial recognition implemented on host application.

5. DELIVERABLES

This project resulted in three major deliverables: a custom video-streaming board which connects to the Crazyflie, the firmware which drives the custom board, and a user application to display the video stream. All three of these deliverables (each presented individually in subcategories of

Section 4) shall be presented as a complete package to Bitcraze, one of our sponsors, for their evaluation and decision on whether or not they would like to partner together to sell this project as an addition to their Crazyflie platform. It should be known that there are design improvements and ideas that have arisen over the course of this project that should be addressed before fully releasing this project as a product. Those improvements and ideas are presented in Section 6 below.

6. RESULTS AND FINAL EVALUATION

The final version of this project allows for a 400x300 video stream at approximately 25 FPS when in bright lighting conditions from multiple simultaneous streams. While there is a theoretical max of 256 simultaneous streams as discussed in Section 4.2.3, there may exist a more realistic ceiling in the number of simultaneous streams that depends on the amount of external traffic on the LAN in use and the processing speed of the host computer in use. A maximum of five simultaneous streams was tested simply due to a lack of components and test equipment. There was no clear degradation of video quality when streaming from the devices simultaneously and any limit on the amount of boards streaming at once would likely be due to network bandwidth.

An image from the Crazyflie in flight can be seen in Figure 22. The blurriness is a result of camera vibrations due to the absence of any mechanical isolation between the camera module and the chassis of the quadcopter. This issue was an unforeseen in the original design of the system. Despite this, the image is clear enough to distinguish features of a face and surrounding environment. Future iterations of the system would improve upon the mechanical subsystem outlined in Section 4.1.6 to include some form of mechanical damping that would effectively reduce vibrations and thus sharpen the images taken from the quadcopter in flight.

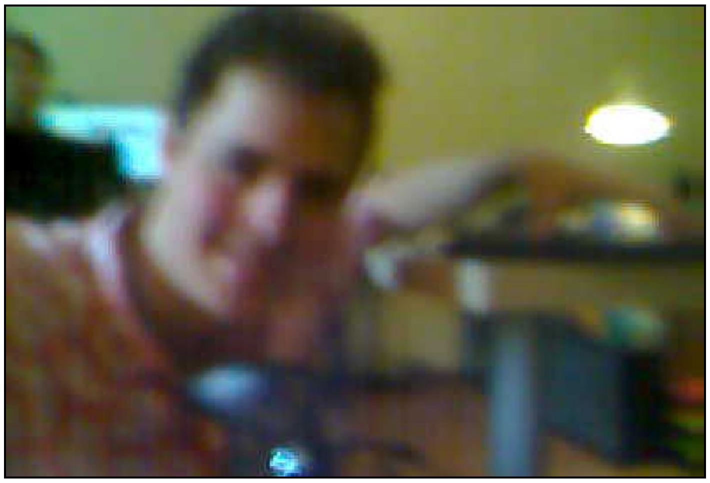


Figure 22: Example picture of the blurry camera capture while in flight.

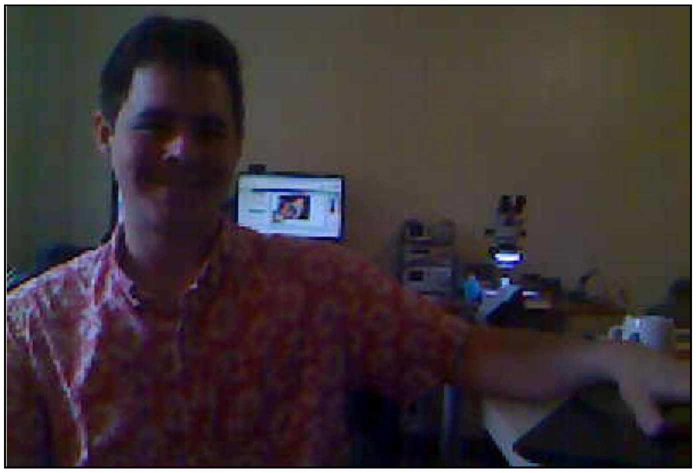


Figure 23: Example of a non-blurry camera capture while not in flight.

As it stands, the overall cost of this project is largely dependant on the cost of the CC3220, as the price per board tends to converge on the per unit cost of the CC3220. For example, it would cost $44.25 to procure and assemble a single board, whereas the price would decrease to $32.92 per board if 100 boards are built simultaneously. That comes out to a 25.6% decrease in per board cost at still low quantity levels. If a small batch order of 1000 boards were to be procured and assembled, the per board cost would fall to $29.09. The results that were achieved over the course of this project and the overall low cost for assembling these boards make this project very appealing to continue to develop to see if even better results and per board costs can be achieved.

7. FUTURE WORK

While the video streaming system successfully achieved performance sufficiently close to what was outlined in the preliminary design goals, there is still a lot of room for improvement. The most glaring limitation is the maximum acceptable frequency of the pixel clock on the CC3220. This bottlenecks the amount of data that can be read from a camera, setting a limit on the frame rate and resolution. In order to fix this, the CC3220 will need to be replaced by an IC that can handle higher data rates. The CC3220 was appropriate for this project because it supports WiFi connectivity and has a parallel camera interface. However, there are certainly other chipsets that provide similar functionality with added performance. Separating the functionality of the CC3220 into two or three other ICs will likely result in a lower cost and better performing product.

In addition to fixing the camera data bottleneck, future revisions of this project will need to use a newer image sensor. The MT9D111 is more than a decade old and is no longer manufactured or supported. Thus, the whole camera pipeline must be redesigned with newer hardware. Ideally, camera manufacturers will see this project as a successful proof of concept and make their newer hardware available for use in future iterations of this product.

In addition to hardware updates, the mechanical design of the system must be improved. More specifically, the mounting of the camera must be mechanically isolated so that the image blur exhibited in Figure 22 is completely damped. This was impossible with the current design as the ribbon cable connecting the camera to the main board was too short. A future design will incorporate some sort of rubber material to dampen the vibrations and allow for a sharper image to be captured.

8. CONCLUSION

Overall, this project was a success with the final product coming very close to the initial design specifications. The project definitely met the goal of being a finished product that could be marketed. In addition, it was sufficiently complex to impress fellow engineers and members of the public alike. The project was in line with the CPE curriculum, consisting of a mix of complex hardware and software subsystems. Future work in digital video streaming technology for nano quadcopters can be motivated and informed by the results achieved in this project.

TABLE OF ABBREVIATIONS

BAC	Bay Area Circuits
CPE	Computer Engineering
CSI	Camera Serial Interface
FPS	Frames Per Second
IC	Integrated Circuit
IP	Internet Protocol
I2C	Inter-Integrated Circuit
LAN	Local Area Network
MCU	Microcontroller Unit
OSH	Open Source Hardware
PCB	Printed Circuit Board
PLL	Phase-Locked Loop
SAC305	96.5% Sn, 3.0% Ag, 0.5% Cu
SiP	System in Package
SMT	Surface Mount Technology
TI	Texas Instruments
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
WLAN	Wireless Local Area Network

APPENDICES

Over the course of this project, design decisions were made and meaningful points of discussion occurred with regards to how the project should be designed and worked forward. A live-copy of the design notebook that exists as a record of these decisions and discussions can be found here.

REFERENCES

[1] "The Astounding Athletic Power of Quadcopters." Performance by Raffaello D'Andrea, TED: Ideas Worth Spreading, *TED Conferences, LLC*, June 2013, www.ted.com/talks/raffaello_d_andrea_the_astounding_athletic_power_of_quadcopters?utm_campaign=tedspread&utm_medium=referral&utm_source=tedcomshare.

[2] "Crazyflie 2.0." *Bitcraze,* Bitcraze AB, www.bitcraze.io/crazyflie-2/.

[3] "CC3220 SimpleLink™ Wi-Fi and Internet of Things - Technical Reference Manual." *Texas Instruments*, http://www.ti.com/lit/ug/swru465/swru465.pdf

[4] "Bitcraze Wiki." *Projects:crazyflie2:Architecture:Index [Bitcraze Wiki]*, https://wiki.bitcraze.io/projects:crazyflie2:expansionboards:index

[5] "CMOS MT9D111 AF Camera Module." *Aptina Imaging*, March 2013, http://www.uctronics.com/download/MT9D111_AF_Module_ug.pdf

[6] "1/3.2-Inch System-On-A-Chip (SOC) CMOS Digital Image Sensor - MT9D111." *Micron Technology Inc.*, https://www.uctronics.com/download/cam_module/MT9D111_SOC2010DS.pdf

[7] "Bitcraze Wiki." *Misc:investigations:thrust [Bitcraze Wiki]*, https://wiki.bitcraze.io/misc:investigations:thrust

[8] "Capabilities." *Bay Area Circuits,* 25 Apr. 2018, https://bayareacircuits.com/capabilities/

[9] "Chip Quick SMD291SNL350T3 - Solder Paste No-Clean SAC305 in Jar 250g T3 Mesh." *Chip Quik*, 2017, http://www.chipquik.com/datasheets/SMD291SNL250T3.pdf

[10] "CC3220MODx and CC3220MODAx SimpleLink™ Wi-Fi CERTIFIED™ Wireless MCU Modules." *Texas Instruments*, http://www.ti.com/lit/ds/symlink/cc3220mod.pdf