SELF ADJUSTING REAR VIEW MIRROR

By

Adam Magier (magier2)

Derek Wood (drwood2)

Thomas Jarosz (tjarosz2)

Final Report for ECE 445, Senior Design, Fall 2018

TA: Anthony Caton (caton2)

12 December 2018

Project No. 22

Abstract

Our project goals were to create a self-adjusting rearview mirror which would use a facial-recognition neural network to maintain a constant fixed perspective for the user. Our project failed to fully realize this vision, yet it still provides a proof-of-concept for the initial design.

At present, the project is capable of controlling all three motors and tracking a user via a face-tracking algorithm. The system is powered using a 3.3V laboratory power supply, not the intended 12V to 3.3V DC-DC converters, and the motor microcontroller unit was replaced with a standard Arduino due to difficulties with the embedded chip's bootloader.

The current design did not fully realize our precision targets, due to unexpected hardware problems which caused motor slippage and lack of a full hardware set. Achieving the original project goals would require further software debugging, as well as a hardware redesign for smoother, more precise operation.

Contents

1. Introduction

Car technology has rapidly advanced in the past few decades, with the advent of numerous safety features and driver aids designed to improve focus and reduce distractions. Such advancements have yet to reach the rearview mirror in any significant quantities. As of now, current improvements of the rearview mirror involve using a digital display to feed input from a rear-mounted camera to enhance the view through the rear of the vehicle[1]. Such designs have a number of significant advantages over the traditional rearview mirror, but the cost of including a rear-facing camera and LCD display has led to slow adoption[2].

This project is a proposed alternative to this advancement, which offers an improvement on the traditional rearview mirror without as significant a change in the user experience as these newfangled digital display mirrors, at a lower price point. Rather than using a camera and display to maintain a constant, clear reaview, it is instead possible to physically move a mirror to maintain a clear, constant view.

Building a self-adjusting rearview mirror required solving several engineering problems, both hardware and software. Primarily, a control system must use information from a facial-recognition program in order to physically manipulate some sort of mirror. For simplicity's sake, the control system should receive a single point of focus from the facial recognition program, and it should compute movement angles to maintain focus on this single point using as simple an approach as possible.

Through effective hardware design, the facial recognition program can act as an observer from the direct center of the mirror, which then simplifies the control logic significantly. This control logic can then direct the hardware to rotate around the fixed observation point, until further adjustments are necessary. A camera can provide input to the facial recognition program, which can then output data to a microcontroller. This microcontroller can then control the motorized hardware which the camera is mounted to, in order to maintain an illusion of a fixed view even as the user shifts position.

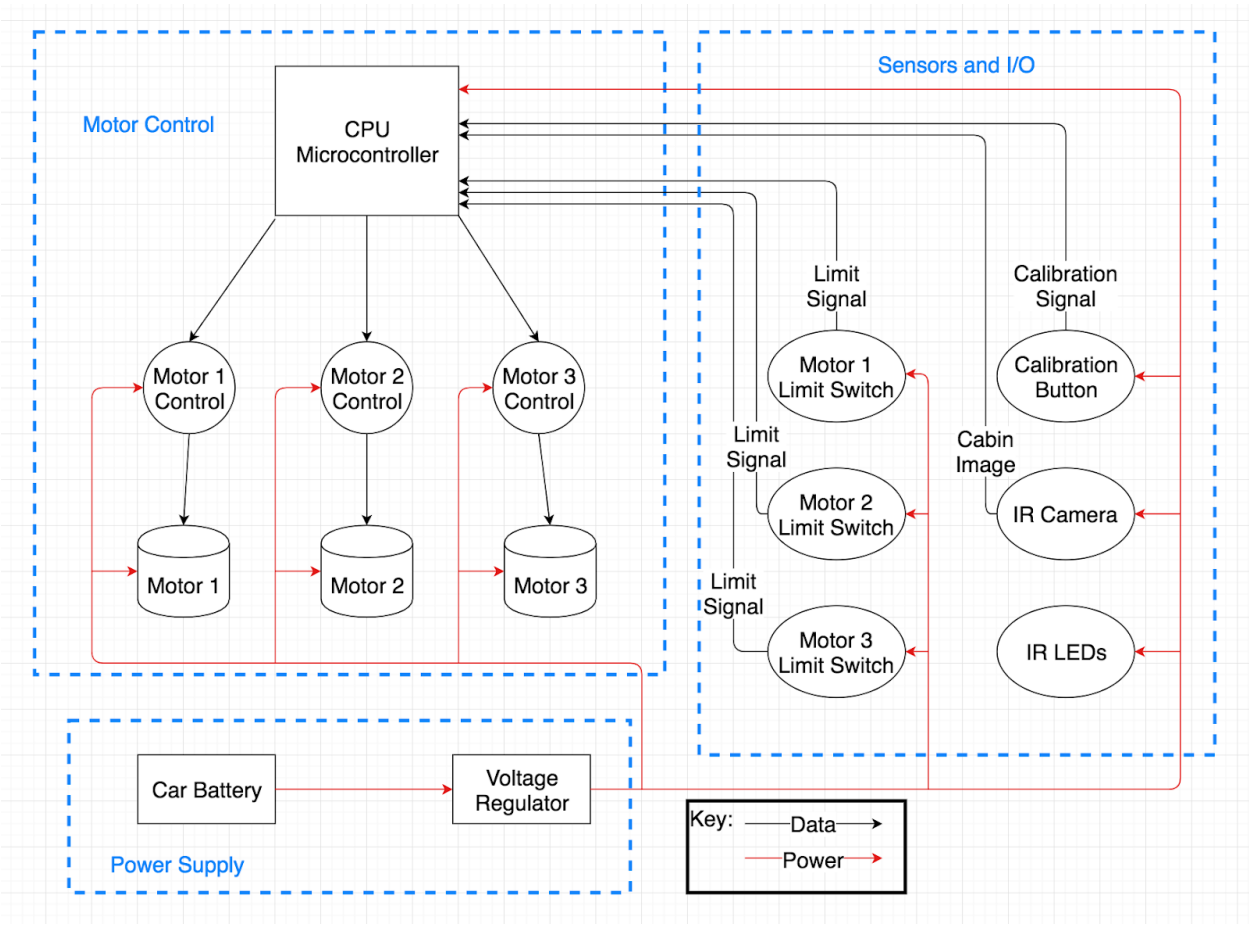


Figure 1: Full Block Diagram

2 Design

2.1 Mechanics

The completed project can be broken down into Power, Control, and Sensor components. The power supply would power the rest of the components, and the Sensors component provides feedback for the microcontroller. The microcontroller then provides signals to the motors based on the limit switch data, infrared camera data, and calibration button data.

2.1.1 Mount

The mount itself was designed to allow for rotation along each of the three rotational axes while allowing for the center of both the camera and the mirror to be as close to the intersection of the three axes as possible. The reason for having these distances be as close to zero as possible is to minimize or even eliminate the need for additional calculations in the form of translation matrices. It is also possible to achieve the same functionality using only two rotational axes, however it was one of our main goals to use all three rotational axes.

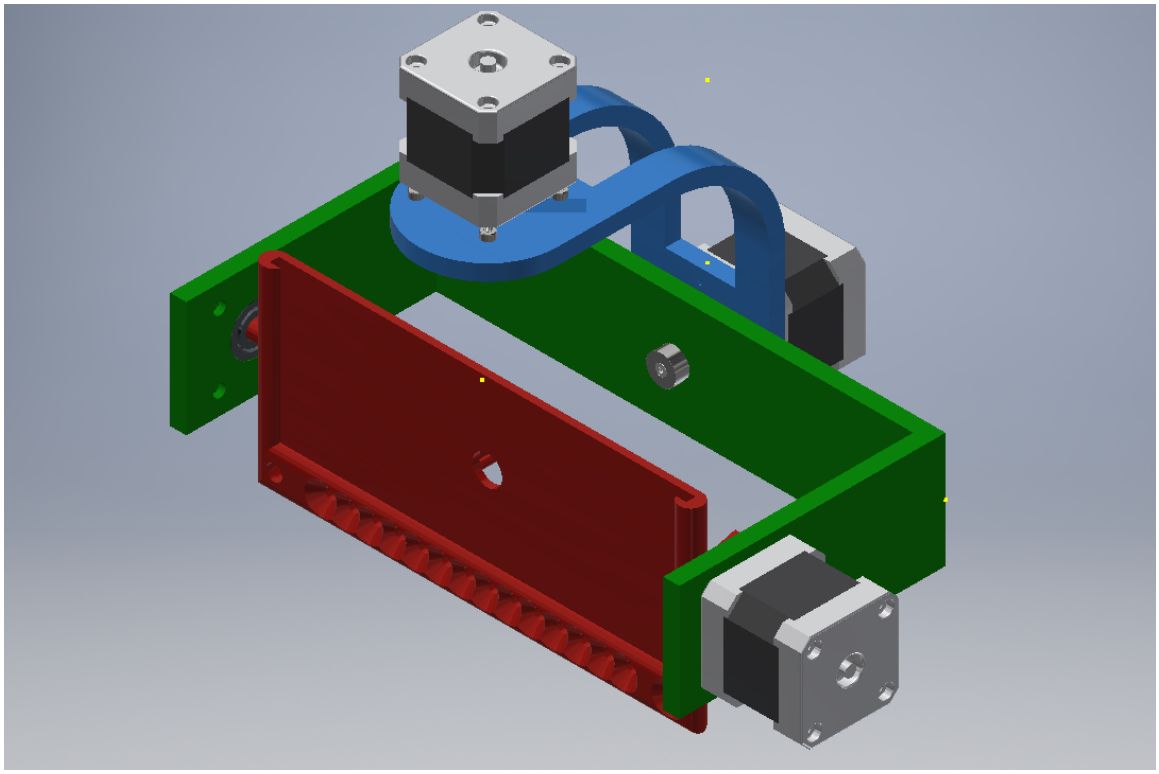


Figure 2: Final Camera Mount

The final design was heavily inspired by a camera gimbal, which is a tool used in videography to stabilize video footage. A camera gimbal places the camera right in the center of a mechanism which allows for

rotation along each of the three rotational axes, which is exactly what we wanted. The mechanism was designed in Autodesk Inventor and constructed from FDM printed PLA+ plastic. Different materials were considered for constructing the mounting mechanism, such as machined aluminum, however after taking into consideration turnaround times, ease of use, and weight, we decided to settle on using PLA+. The only drawback in using the plastic was the lack of rigidity, however the benefits far outweighed the detriments.

2.1.2 Stepper Motors

The motors that were used in this design were stepper motors. The reason why stepper motors were chosen was because of the high torque and precise orientation controladded circuit complexity of designing and implementing dedicated driver circuits for the stepper motors. The downside to using stepper motors is that they have no built in way of measuring absolute orientation, so it's necessary to implement additional hardware to keep track of the orientation of the motor. The initial design called for servo motors, which would have alleviated the need for additional hardware for orientation, however it would significantly reduce the complexity of the circuit.

2.1.3 Limit Switches

The limit switches were added to determine absolute orientation in the software. As mentioned in the subsection for the stepper motors, the decision to use stepper motors necessitated additional hardware to determine absolute orientation. Limit switches and rotary encoders were the largest considerations for additional hardware, however the ease of implementation with limit switches ended up being the deciding factor. The downside of using limit switches is that in the event that a motor slips, it's necessary to calibrate the mount by hitting the limit switches and re-determining the absolute orientation.

2.1.4 Calibration Button

The main purpose of the calibration button is to allow the user to set a custom orientation for the mirror that the system will maintain during operation. This was one of the main features that was planned for implementing in the overall system since many drivers have personal preference regarding mirror orientation. The main alternative to a button that was considered was a capacitive touch sensor, however given cost and ease of implementation a button became the obvious choice.

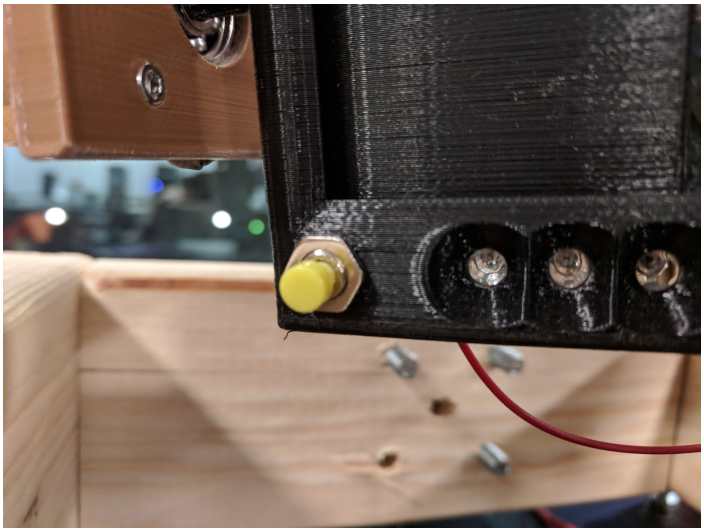


Figure 3: Photograph of button placed onto mount

The specific button chosen had to meet two main requirements: it had to be panel-mountable, and it had to be easy to press. The button we ended up choosing was sourced through Sparkfun, specifically so that we could get free shipping on our order, but also because it fulfilled all of our requirements.

2.1.5 Mirror

In order to place the IR camera at the intersection of the three axis of rotation of the camera, the camera needed to be located at the direct center of the mirror. For aesthetic reasons, it was decided that the camera would be in some way hidden behind the mirror, so as not to impact the visibility of the mirror or the user experience for the driver. Given the selected camera, which functions in both the IR and visible light wavelengths, the clearest option for concealing the mirror was to use the principle of a 2-way mirror.

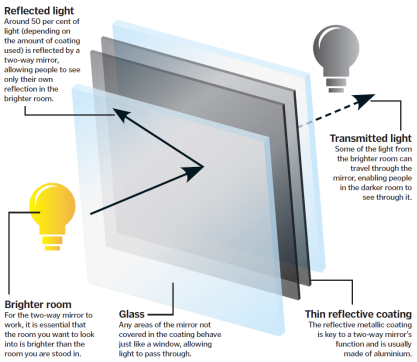


Figure 4: Principles of a two-way mirror[3]

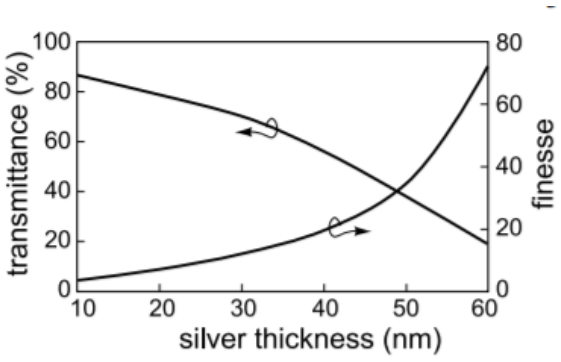


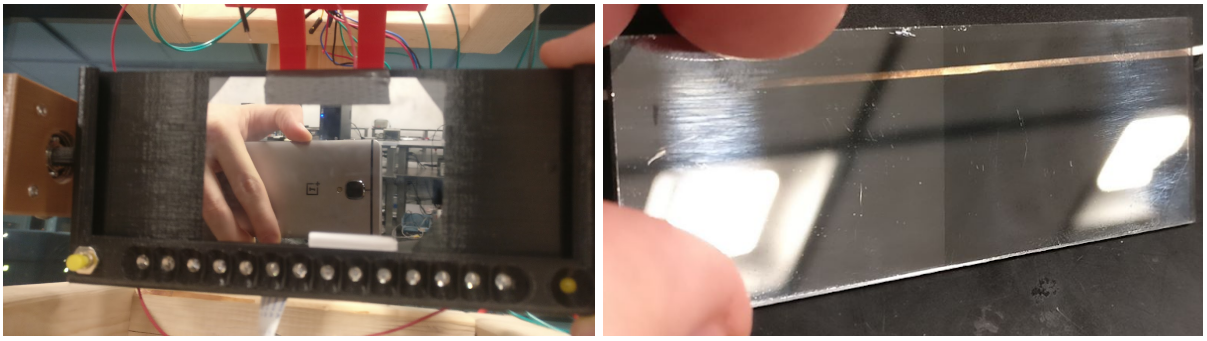
Figure 5: Thickness vs Reflectance and Transmittance of Silver films[4]

Using an E-Beam evaporator, it was possible to deposit various thicknesses of Germanium metal onto glass substrates, allowing for experimental testing of various film thicknesses.

A literature search proved unhelpful in determining an appropriate film thickness specific to Germanium. However, there was rich variety of sources for the more common materials; Aluminum and Silver[5]. These values typically ranged in the tens of nanometers, which was used as a benchmark for testing experimental thicknesses.

Thickness	Transmittance
0nm	Transparent
7.5nm	Mostly transparent
12.5nm	Slightly transparent
20nm	Mostly reflective

Table 1: Material thickness versus Transmittance



Figures 6 and 7: 20 nm final mirror, as well as glass slide with 12nm and 7.5nm thicknesses

2.2 Electronics

2.2.1 Microcontroller Unit (MCU)

The microcontroller needed to have enough digital input and output pins to fully support the control of the three motors and the peripheral features (limit switches, etc). The standard for MCU are the ATmega branded line of MCUs, and with the many varieties it was very easy to find a suitable MCU. The MCU that was ultimately chosen was the ATmgea32u4. The main reason for choosing this chip over a different kind of MCU was the built-in USB interpreter, so it wasn't necessary to have additional hardware to convert the USB signal into serial signals.

2.2.2 Stepper Motor Driver

The selected stepper motors require a separate driver in order to properly manage the control and power pins. A TI-sourced DRV8834 fulfilled these purposed and was fully functional in the final design.

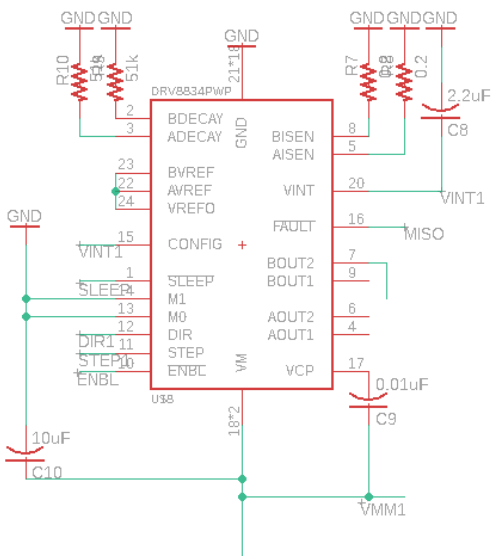


Figure 8: Motor driver schematic

All peripheral components were selected in accordance with the provided datasheet. The only significant changes was swapping the final sense resistors for 0.3Ω. The initial selection was 0.2Ω. This increase in resistance decreased the motor supply current to roughly 1.2A which is below the motor's 1.7A rated current, but this was preferable to the 0.2Ω resistor, which would have provided 2.0A. This would have been above the 1.7A rating.

Any comparable stepper-motor driver would have provided similar functionality.

2.2.3 Power Circuitry

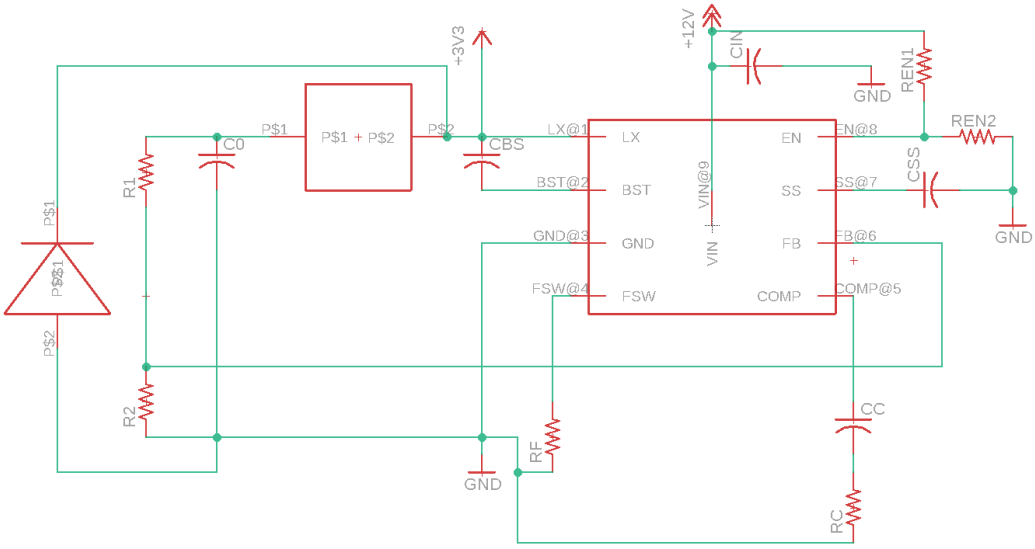
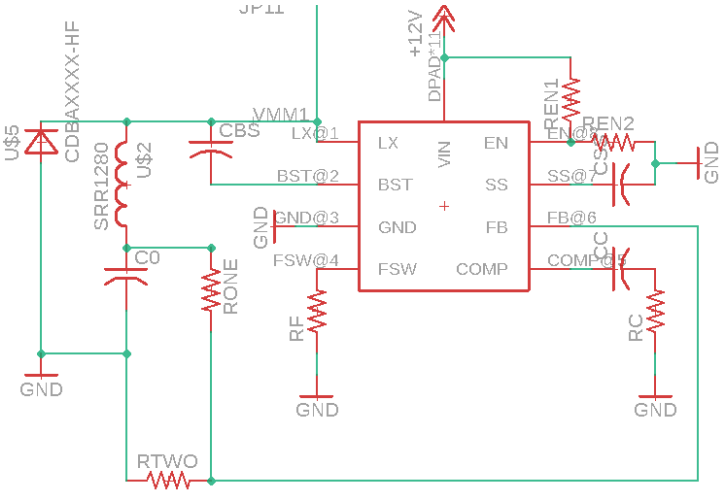
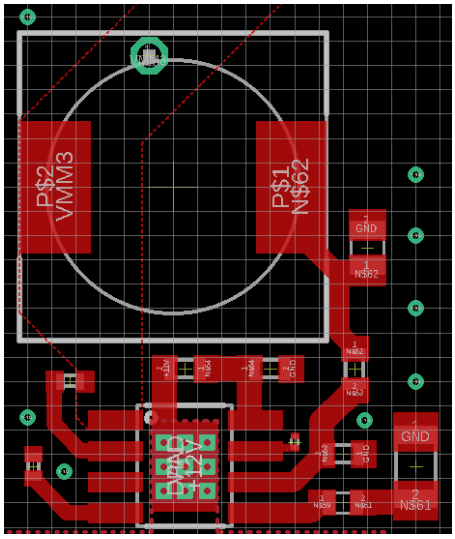


Figure 9: Voltage step down circuit

The motors were intended to be powered by a set of three buck voltage regulators which would each power one of the motor drivers. Buck regulators were the most appropriate implementation for providing a DC-DC step down, due to the simple design and moderately high efficiency. A linear voltage regulator would have been impractical due to the significantly lower efficiency, given the power requirements involved.





Figures 10 and 11: Final regulator schematic and Layout

All part selection was based off of the equations provided within the AOZ1284's datasheet. For the final board, most of the parts were re-selected for the sake of convenience. Many of the initial parts were 0201-sized, which is a challenge to solder if nothing else. The final fabrication was greatly simplified by using 0402, 0604, and 0805 parts instead.

2.2.4 Central Processing Unit (CPU)

The CPU is meant to be an abstraction of the central computing system in a production car. A modern car infotainment system typically uses a dual or quad core CPU coupled to a reasonably high-powered GPU. The Raspberry Pi which was used for facial recognition was actually underpowered in comparison[6].

2.2.5 Infrared Camera

The IR camera used was chosen due to its compatibility with the Raspberry Pi used for facial recognition. Any similar IR camera could have served the same purpose. One disadvantage of the selected camera was the narrow field of view (26.67 degrees). This limited field of view was sufficient for demonstration purposes, but a wider-lensed camera would have been more appropriate.

2.3 Software

The software running on the MCU and CPU are the brains of our project. The software is responsible for processing camera data and determining where and when to move the mirror. The two major software components are the eye detection and control logic software. These two components are discussed below.

2.3.1 Eye Detection

Eye detection software is used to determine the location of the driver's eyes in images. The driver's eye location is used to determine where the mirror should adjust to, so accurate eye detection is necessary for accurate mirror adjustments. There are two steps to finding the location of the driver's eye. First, our software must find the head of the driver. The software must then find the location of the eyes on the driver's face.

To find the driver's head, a tensorflow based Single Shot multiBox Detector (SSD) was used [8]. This particular model was chosen because it is very robust; it can identify faces in varying orientations, light settings, and is not affected by hats and glasses. The downside of using this model is the memory consumption (300 megabytes) and computation time required for classification. Identifying the faces in one image (of size 1280x720p) takes upwards of 10 seconds to run on our Raspberry Pi. As we set a requirement of processing 1 image per second, this rate does not suffice. In general, the computation time required to classify an image includes a fixed time cost of loading model and image information, and a variable time cost proportional to the size of the image. To decrease the fixed time cost per image, we process four images at a time. This is done by making a collage of four images, and passing this image into the classifier. To decrease the variable time cost per image, we decrease the height and width of the image by a factor of two. We also only use the left half of each image, as we assume the driver should never be to the right of the mirror. Each resulting image is one eighth of its original size. By implementing these two optimizations, we were able to decrease the processing time from about 10 seconds per image to about 1 second per image, satisfying our requirement. Shown below is a graph of images processed versus time.

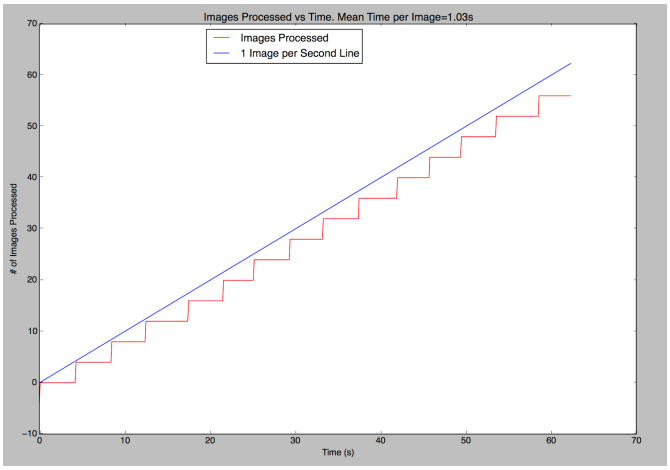


Figure 12: Graph of images processed (x axis) versus time (y axis)

While this software can identify all faces in an image, we do not want to know the location of all faces, but only the location of the driver's face. To identify the driver's face, we filter out all faces which are not above a certain size limit (3.5% of the full image). If there is more than one face larger than this threshold, the face farthest to the left is chosen to be the driver's face. Intuitively, the logic behind these decisions is that the driver will typically be closer to the mirror than anyone else in the car, so the driver's head will be the largest in the image. The driver is also expected to be the face farthest to the left in the image. The head size threshold was determined through experimentation: the driver's head size is typically around 6.5% of the image, while a passenger's head size is around 1.0% of the image.

With the bounding box of the driver's face in the image given, the next task for our software is identifying eyes on the driver's face. To achieve this, we use a Haar feature based cascade classifier [7]. This classifier was specifically chosen because it is fairly robust when detecting eyes, and is very lightweight. Specifically, the eye detection model is 3 orders of magnitude smaller than our face detection model (300 kilobytes vs 300 megabytes). While this eye detection algorithm performed well in terms of precision (3.1), it struggled in terms of recall (3.2).

Loading...

Loading...

Where *TP* is true positives, *FP* is false positives, and *FN* is false negatives.

To filter out the false positives made by this classifier, several techniques were used. A major issue for us was noses; nostrils were often incorrectly labeled as eyes. To prevent this, we only ran eye detection on the top half the the driver's face bounding box (this also helped improve runtime). Often, two eye bounding boxes were given for the same eye. To fix this behavior, if it is found that the intersection of any two bounding boxes is larger than half the size of either bounding box, the larger of the two bounding boxes is disposed. Finally, as it became apparent that false positives were more common to appear below true positives in the image, the two bounding boxes farthest up on the driver's face were used as the bounding boxes for the driver's eyes. The average of the center of these two bounding boxes is used as the location of the driver's eyes.

In some cases, although the driver's face gets detected, the driver's eyes do not get detected. This is very common behavior of the software when the driver is wearing sunglasses. As we specifically stated our mirror should be able to track the user's eyes when wearing sunglasses, we had to find a way to estimate the user's eye location given only the head location. We found an effective way to make this estimate is to assume the driver's eyes are located in the middle of the face bounding box, one quarter away from the top of the face.

2.3.2 Control Logic

Using eye detection, and given the current orientation of the mirror and a reliable mapping from pixel location to angle from camera, the absolute position of the user's eyes can be determined. Depending on the state of the control logic, the position of the user's eyes is used in different ways.

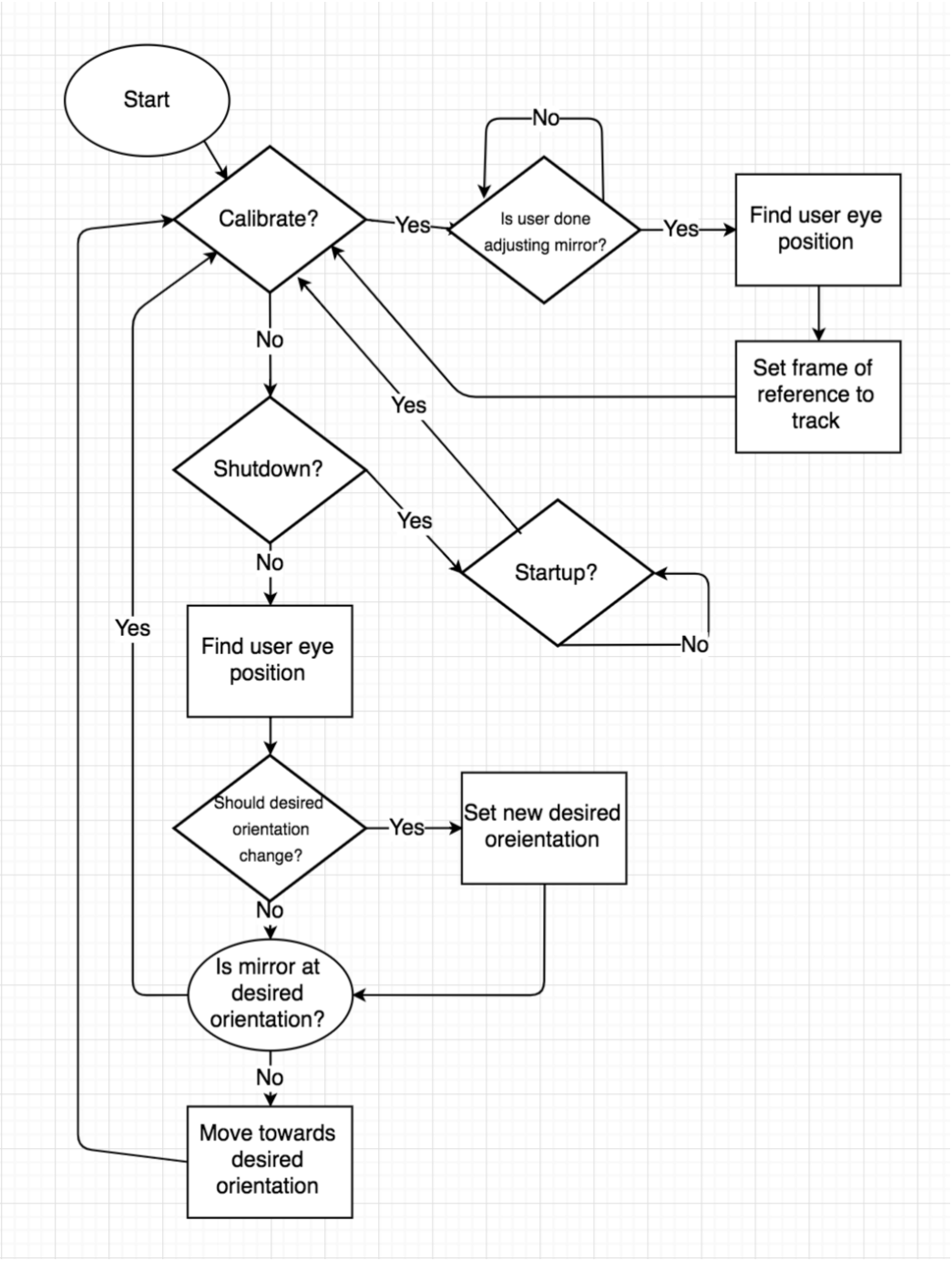


Figure 13: Software flow chart

From the flow chart above, you can see that the user eye position is either used to determine where the mirror should move, or where the rear window is located. The user's input (via pressing the button) determines how the user's eye position should be used. A single click of the button indicates that the user wants to set a new rear window location. If there is no click, the mirror should continue adjusting itself. Additionally, if the user holds the button for more than five seconds, the system shuts off. It is the

MCU's responsibility to process the button and limit switches, along with physically controlling the motors. It is the CPU's responsibility to process images, make computations regarding different orientations, and implement the flowchart shown above. THe MCU and CPU communicate serially; all communication between the CPU and MCU is initiated by the CPU. Specifically, a four byte command is written to the MCU; the MCU then responds with 4 bytes of information after executing the command. The CPU can tell the MCU to rotate the motors a specific amount of steps, rotate the motors until they hit their limit switches, or return information regarding whether the user has pressed the button.

3. Design Verification

3.1 Mechanics

Everything for the mechanics of the system was verified and worked as expected, however upon full construction the motors were not functioning as expected. This was not a fault of the motors however, and was moreso a consequence of having an inadequate power supply from the circuit. When all three motors would try to step in quick succession, the power spike would be too much for the bench power supply that we were using, and as such at least one motor would have the potential to skip. Individually, each of the motors were capable of functioning as expected.

3.2 Electronics

The two main features of the electronics that weren't able to pass verification were the MCU circuit and the power supply circuit. It appears that the MCU that ships from the manufacturer comes with a certain bootloader, which is meant to be programmable directly from shipping. However we encountered severe errors upon attempting to program the MCU. Comparing the schematic of our implementation to the schematics of known implementations of the same MCU, everything is more or less identical.

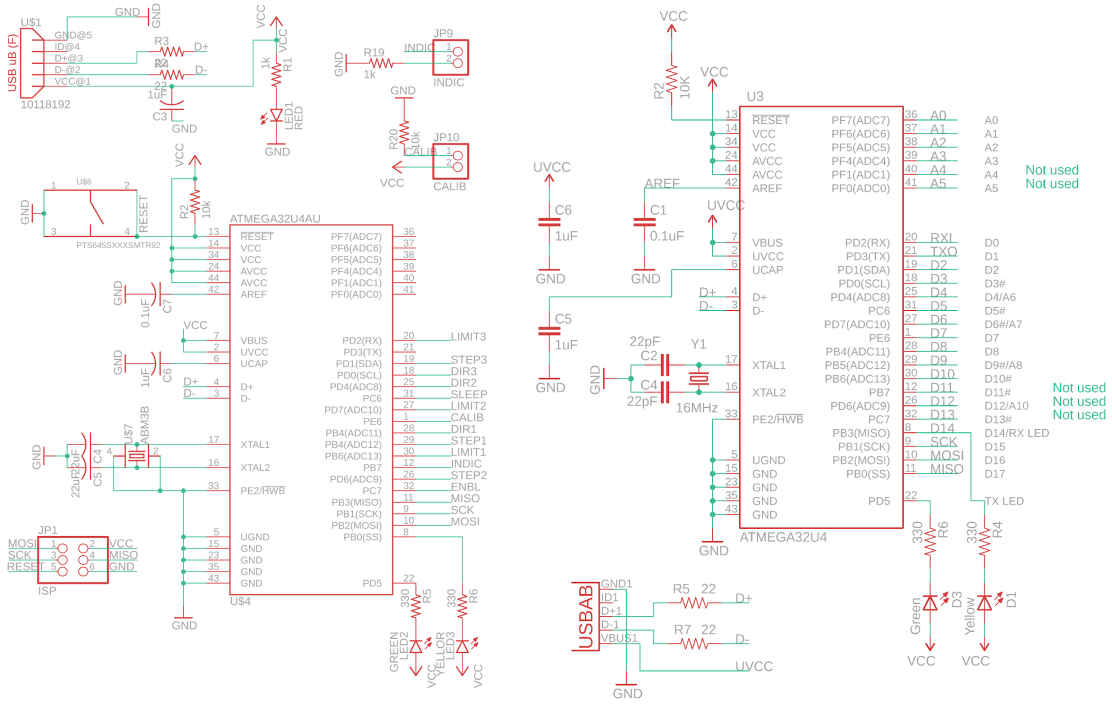


Figure 14: Designed microcontroller schematic Figure 15: Reference Sparkfun schematic

Regarding the power supply circuit, the main issue seemed to stem from the fact that there was a lot of power going through the system at a high frequency, and as such the amount of heat generated by the chip was too high for it to handle. Connecting the power supply circuit up to a bench power supply at 12V would heat up and throttle within 30s, which was not the behavior expected from the datasheet.

Upon further investigation the switching frequency was too high (1 MHz) and as such the inductor could have been effectively a resistor and did not exhibit the characteristics for the conversion circuit. Due to our lack of power systems knowledge, we could not fully and effectively diagnose this issue.

3.3 Software

The software passed all facets of verification. While we did have a software implementation that processed each image in approximately 0.9 seconds (via further image descaling), we felt the quality of classification for the slightly slower implementation outweighed the speed of the quicker implementation. The final implementation processed images at a slightly slower rate, 1.03 seconds per image.

4. Costs

4.1 Parts

Table 3: Itemized cost of parts per assembly

Digikey Part Number	Part	Qty	Price	Bulk
1080-1032-ND	IR LED	2	$0.50	$0.14
1568-1106-ND	Stepper Motor	3	$16.95	$16.95
609-4613-1-ND	USB Connector	1	$0.45	$0.29
ATMEGA32U4RC-AU-ND	MCU	1	$4.10	$3.50
296-41246-1-ND	Stepper Motor Driver	3	$2.76	$1.30
641-1707-1-ND	Diode	3	$0.49	$0.09
1276-6780-1-ND	Capacitor	3	$0.38	$0.08
587-5296-1-ND	Capacitor	3	$0.10	$0.01
RNCP1206FTD604RCT-ND	Resistor	3	$0.10	$0.01
RR08P31.6KDCT	Resistor	6	$0.11	$0.01
RR08P10.0KDCT-ND	Resistor	6	$0.11	$0.02
785-1689-1-ND	Voltage Regulator	3	$1.06	$0.42
1276-6698-1-ND	Capacitor	5	$0.42	$0.08
P10864S-ND	Limit Switch	3	$0.38	$0.20
CSR0805FKR200CT-ND	Resistor	6	$0.42	$0.05
RNCP0805FTD10K0CT-ND	Resistor	4	$0.10	$0.01
732-4981-1-ND	MCU Status LED	1	$0.15	$0.10
732-4978-1-ND	MCU Status LED	1	$0.15	$0.10
732-4971-1-ND	MCU Status LED	1	$0.15	$0.10
CKN9084CT-ND	Switch	1	$0.17	$0.11
1276-6524-1-ND	Capacitor	1	$0.10	$0.01
1276-1739-1-ND	Capacitor	1	$0.10	$0.01
311-47.0KLRCT-ND	Resistor	3	$0.10	$0.00
732-7751-1-ND	Capacitor	3	$0.10	$0.02
RMCF1206FT51K0CT-ND	Resistor	6	$0.10	$0.00
RMCF0603FT330RCT-ND	Resistor	2	$0.10	$0.00
RMCF0603FG1K00CT-ND	Resistor	3	$0.10	$0.00
SRR1280-220MCT-ND	Inductor	3	$1.27	$0.47
		Total:	$84.61	$63.81

From analyzing the costs of purchasing the parts at retail and in bulk, the savings are approximately $20 per assembly. This does not seem like much savings, and is definitely far lower than what's expected from buying in bulk. Looking at the price breakdown it's clear that the motors contribute the majority of the cost, with three motors costing just over $50. Once the motors are removed from the equation, the cost difference shrinks by a factor of almost ⅓, as opposed to just $20.

4.2 Labor

The labor costs will be analyzed in two different manners - the total cost of researching and developing the mounting system, as well as the cost of building a given assembly. The costs will be determined as a function:

$$hourly\ rate \times hours \times 2.5$$

For the research and development, we are assuming that we spent at least 500 hours coming up with the entire system, whereas for building a given assembly we are estimating that it should take no more than 30 hours to fabricate all the parts necessary and put everything together. We are also assuming that we are working as decently paid machinists, at an hourly rate of $25/hr. The costs therefore come out to:

$$25 \times 500 \times 2.5 = \$31250$$

$$25 \times 30 \times 2.5 = \$1875$$

for R&D and assembly, respectively.

5. Conclusion

5.1 Accomplishments

The final product was capable of rudimentary facial tracking and orientation. Though performance was suboptimal, the project was a technical success.

5.2 Uncertainties

The hardware and software failures were never fully diagnosed and fixed. It is impossible to say with certainty that our proposed solutions would result in a fully functional product.

5.3 Ethical considerations

There are several potential safety hazards and ethical concerns with our project. The mirror will be used to safely control a vehicle, and the user will be constantly observed by a camera. Any failures on the part of the mirror could result in the loss of life not only the user, but anyone surrounding the user's vehicle, as well as loss of the user's privacy.

Any collected visual data is only transient and will not exist once the computations have been completed. This data cannot be stored by the system, and the hardware does not have any ability to transmit visual information to the outside world. A malicious agent would need physical access to the hardware and would need to make significant modifications to the hardware to access, store, or transmit any data collected by the system. The level of access and effort required for such an endeavor renders additional mitigation strategies pointless. With the necessary level of dedication and resources required for such an attack, an attacker would have ample other opportunities to violate the user's privacy.

Additionally, there may be unknown problems caused by quirks in human perception or the operation of the hardware and control system. If movement is jittery, too frequent, or too infrequent, the mirror may be uncomfortable to use, or may not function as well as a traditional mirror.

In designing this project, we took into consideration the IEEE Code of Ethics[10]. We believe our design sufficiently adheres to the code, especially with regards to #1: "to hold paramount the safety, health, and welfare of the public". Our project was motivated by a desire to protect the health and safety of the public, and the final design must produce a net gain in safety to the user, who is realistically trusting the system with their lives.

5.4 Future work

When considering future work for our project, there are improvements that can be made on both the hardware and software side.

5.4.1 Hardware Improvements

A major source of failure for our project was stepper motor slipping. Our current motors are very prone to slipping, causing the actual orientation of the mirror to differ significantly from the expected orientation of the mirror. Without knowledge of the actual orientation of the mirror, the mirror cannot be properly adjusted. There are several things we could do to resolve this issue. Acquiring motors with higher torque would decrease the likelihood of slipping. Additionally, motors with D-shafts would prevent the mirror mount from slipping from the motor shaft. By adding rotary encoders to our project, we can achieve closed loop feedback for the orientation of our mirror, and correct for any slipping that may occur.

Vibration was also a major issue for our project. The motors stepped at a given frequency, which would cause the mount to visibly vibrate. In extreme cases, the vibrations of one motor rotating would even cause another motor to slip. To resolve this issue, we could build the mirror mount out of more rigid material.

During our demo, the PCB was a rat's nest of wires. This was because the onboard CPU and voltage regulator were not working, so off board devices had to be wired in. Fixing these components to work on board would significantly improve our project.

5.4.1 Software Improvements

One significant software component that was never implemented was high accuracy camera parameterization. Knowing the mapping between pixel location and angle from camera is very important for computing the position of the driver's eyes. This mapping was going to be produced by stepping the z and y axis motors through every step, and record the location of an LED along the way. THis would give us a mapping accurate to 0.9°. SInce we could not get our motors to move precisely, this mapping could not be generated.

Prior to our demo, we could successfully communicate back and forth between the CPU and MCU. During our demo, however, communication only worked one way (from the CPU to the MCU). THe result of this one way communication is that the user has no way to reset the mirror position or shut off the mirror.

References

[1]Adams, Eric. “CADILLAC’S ALL-SEEING REARVIEW MIRROR PEERS INTO THE FUTURE OF CARS.” Wired, 3 November 2016, https://www.wired.com/2016/11/cadillac-ct6-rearview-mirror-camera

[2]"Panasonic to Start Mass Production of Electronic Rear-view Mirrors, the First Product Jointly Developed with Ficosa” Panasonic. 25 September 2017, https://news.panasonic.com/global/press/data/2017/09/en170925-5/en170925-5.html

[3]”How do two-way mirrors work?” How Stuff Works. 14 December 2015, https://www.howitworksdaily.com/how-do-two-way-mirrors-work

[4]Bartek, M. “Silver-based reflective coatings for micromachined optical filters” Delft University of Technology, Faculty for Information Technology and Systems, Laboratory for Electronic Instrumentation/DIMES, Mekelweg 4, 2628 CD Delft, The Netherlands. 12 February 1999. https://pdfs.semanticscholar.org/b9a3/e1b0aef9cc0892d7d23f2c5924d09ae51477.pdf

[5]Grant, Paul. “The Optical Properties of Thin Germanium Films”Clearing House. June 1965 http://www.w2agz.com/Publications/Theses/Harvard/The%20Optical%20Properties%20of%20Thin%20Germanium%20Films,%20AD%20619%20071%202.pdf

[6]“Qualcomm Snapdragon 820 Automotive Platform.” *Qualcomm*, 31 Oct. 2018, www.qualcomm.com/snapdragon/processors/820-automotive.

[7] Kinsley, H. (2018). Python Programming Tutorials. [online] Pythonprogramming.net. Available at: https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/ [Accessed 6 Nov. 2018].

[8] Yeephycho. “Yeephycho/Tensorflow-Face-Detection.” *GitHub*, 6 July 2018, github.com/yeephycho/tensorflow-face-detection.

Appendix A Requirement and Verification Table

Component	Requirement	Verification	Verification status (Y or N)
Car Battery	Outputs 11-13V	A. Measure open circuit voltage using voltmeter B. Ensure voltage is within 11-13V range C. Charge battery and repeat verification if voltage is out of range	Y
Voltage Regulator	Outputs between 3-3.6V	A. Stall at least 1 motor while hardware is operational B. Verify stepper motor supply voltage remains within 2.7-3.6V range C. Verify control hardware voltage remains within 2.7-3.6V range	N
CPU	Return eye tracking information from 1 image per second	A. Provide CPU with 1 image per second B. Verify CPU computes eye location information for each image	N
CPU	Provide motor control commands while operational	A. Verify CPU also outputs motor control commands when appropriate	Y
Motor Controller	Circuit must provide ample power to the motors	A. Send constant driving pulses to the motor B. Verify motor does not move and circuit does not fry	Y
Motor Controller	Circuit must send out appropriate pulses from CPU input	A. Configure CPU to send pulses to the control circuit B. Verify that the motors move upon receiving the signal	Y
Motor Controller	Circuit must send out	A. Configure CPU to send out exactly 400 pulses	Y

	pulses capable of turning the motors by one step	B. Verify that the stepper motor has undergone one full rotation	
Motor	Motors shall be capable of rotating the mirror without stalling	A. Command mirror to tilt right B. Verify mirror shifts right without stalling	Y
IR Camera	Eyes shall be visible in IR camera images when placed 3 feet from a person's face	A. Position camera 3 feet away from subject B. Take image of subject C. Verify image provides clear picture of subject's eyes	Y
IR LEDs	Current draw must be less than 2.5A	A. Power LED array B. Measure current using an ammeter C. Verify total current draw is less than 2.5A	Y
Mirror Mount	Mount must be capable of holding all components (7 pounds)	A. Populate mount with all necessary hardware (or 7 pounds of weight) B. Verify mount does not fall apart	Y
Mirror Mount	Verify mount can be adjusted without breaking	A. Take completed mount B. Adjust mount C. Verify mount does not fall apart	Y
Mirror Mount	Mount must be capable of providing motion along the three rotational degrees of freedom	A. Assemble mount B. Attempt to rotate along each rotational axis C. Verify mount is capable of rotation in all 3 axis	Y