Kyle English
Erika Willi
James Fregien

Final Report

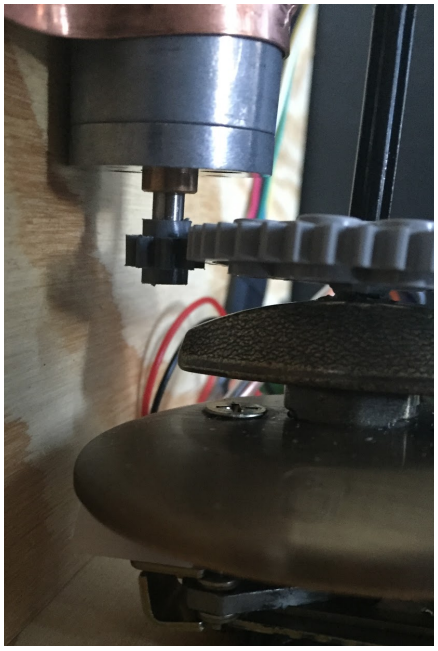
Summary

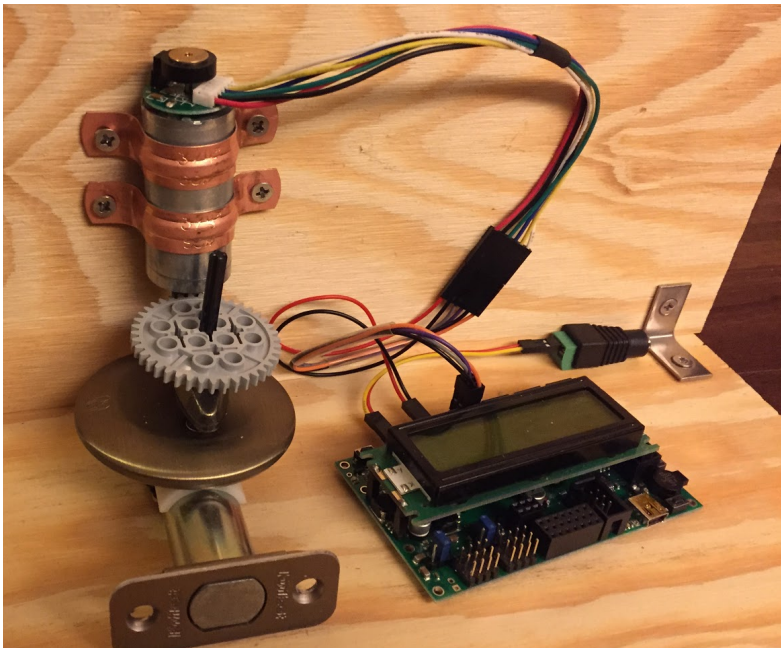
We tried to create the foundation of a home automation hub. The plan was to start with a door lock via Bluetooth and then add a few more home devices once that was complete. Turns out that was plenty of work just to get that one working.

Overview

Mechanics

The mechanics needed to be aligned just right in order for the motor to successfully lock and unlock the door lock. The lock itself was secured to the top of a board for demonstration purposes. A lego axle and gear were then secured to the thumbturn. A perpendicular board was set in place with the motor attached. A small gear was set at the tip of the motor. When the motor turns, the gear at the end of the motor turns, further turning the gear on the thumbturn, which ultimately turn the locking mechanism to lock and unlock the door.





Motor/State Logic

The motor was implemented starting mostly from the motor assignment so we could make the motor turn on command. We didn't implement it as a PID controller for this project, mainly because we wanted more control over its functionality. This was because once we got the motor in place, one mess up and there's a good chance our rig breaks, especially if it overshoots the target. We hardcoded the max speed we wanted to motor to run at based on tests with the rig of what torque it took to turn the lock. We then hard coded the encode value

needed to turn the lock from unlock to a lock position. It ramps up the speed when the command is given (or button is pressed) and then slows down as it nears the target value. Basically, everything is based off of the encode value of the motor which makes this basically a modified P controller. The embedded controller keeps track of whether the state is currently locked or not. For debugging, the top button locks it and the bottom button unlocks it. Also, the +/- of the motor 2 have to be plugged in the way it was configured. If not, then the motor will spin until the encode value rolls over.

Bluetooth

To get started adding Bluetooth to the orangutan boards we first had to research different options for bluetooth modules. We figured the smarter the module we could get, the easier it would be to get our bluetooth connections up and running. From the orangutan website we chose the bluesmirf module with an embedded RN42 chip. There was also a bluesmirf gold module to choose from with an RN41, but we chose the 42 because it only works at a distance of 20 meters compared to the RN41's 100 meters. We wanted the shorter range since we only wanted the chips to be active and connected when in your home. We chose the Bluesmirf silver module with a RN-42 chip to meet our needs, as these chips behave as both master and slave which is great for making a whole network of devices. These chips also have a UART serial connection, which should work well with the Orangutan chips using the UART ports on ports D0 and D1.

To connect to the bluesmirf chips we used the Vcc from the Orangutan board to power the module, and then used a laptop to connect via bluetooth initially. I was then able to send commands to the bluesmirf to place the chip in command mode. In this mode I was able to set the baud rates of the chips and also configure the slave and master properties. To just get it working I initially programmed the chip to a slave mode and connected via my computer. I also wired the Tx and Rx port from the Bluesmirf to the Rx and Tx ports on the Orangutan. In my microcontroller code I set up a serial connection like we have used previously, except instead of connecting to USB_COMM, we connected to UART0. I then sent a string over serial from my microcontroller and was able to get the communication on my computer, but could not figure out how to get communication from my computer back to the Orangutan. After playing with this Bluesmirf for several days, and trying several different configurations of Baud rates and bit parity, I was forced to give up on the Bluesmirf communication. Another group also had trouble getting there Bluesmirf to communicate with the Orangutan chip as well. I decided the Bluesmirf was not going to work properly and to look for other options.

I researched other options and decided to order an HC05 chip, which another group was able to get working. The HC05's had great reviews and many guides to communicate with Arduino boards. I ordered two of the HC05's to hopefully get a small network setup. When I first got my HC05 in, I plugged it into my Orangutan hoping the communication would immediately work. Unfortunately, it didn't, and after playing with the HC05's baud rates and other settings and still being unable to get communications I decided to open the 2nd chip I ordered. After plugging in the second HC05, I was able to get successful bluetooth communication. It turned

out my first chip was dead on arrival. In the end, our project was down to one working Bluetooth module, that we are now using to communicate with the App to control our lock.

Mobile

The mobile app was built with the stock Bluetooth Android components. After pairing the phone to the bluetooth chip embedded on the board, the app is able to create a connection. When the “Unlock Door” button is selected on the app, a command is sent to the chip, triggering the unlocking mechanism. Similarly, when the “Lock Door” button is selected, a command is send to the chip triggering the locking mechanism.

What we learned

We learned not to always assume that when something isn’t working, especially with peripheral devices, to assume that is was our doing. With the amount of errors we encountered trying to set up Bluetooth, a lot of time would have been saved if we could have eliminated the Bluetooth chips immediately. We would have needed more equipment, such as oscilloscopes to quickly determine that the signals being sent from the chip were wrong.

We also learned to always keep problems as simple as possible, and then expand to more complicated solutions as needed. If we would have started with a simple module to start, the HC05, then we would have probably run into less problems trying to get the Bluetooth up and running. Embedded systems are really good at expanding systems, and building complicated solutions, but when you try and start overly complicated it can be hard to narrow in on a single problem

What isn’t working

Bluetooth connections between multiple Orangutan boards. This is because we ran out of working Bluetooth chips to work with, and time.

We also don’t have automatic locking when the Bluetooth app is disconnected because the Bluetooth chips we used had no way to tell a host over UART if a device was disconnected.

The mobile app is currently not fully working. The app will pair with the bluetooth chip and the app will send one command, either to lock or unlock. The app however will not send any further commands. If the app is debugged and stepped through, it will work just fine, so we are not sure how to resolve this issue.

Future Work

Motor/State Logic

A lot more could be done with the motor portion of this project to make it more robust. For instance, detecting if the door isn’t closed all the way and thus unable to lock. Try to lock, then reverse itself if it doesn’t reach it’s destination and throw an error to the user. We could also implement it much more like a PID controller, though we’d have to make it dynamic since door locks have different torque needs. Much of what was done with the motor in this project

was hardcoded for our specific door lock. A lot could be done to improve this to make it more robust and resilient to the type of door lock.

Devices
If we had more time, we also would have loved to have added more home devices such as lights, thermostats, cameras, etc. Lights should have been fairly straight forward so we would have tackled that next.

Bluetooth
With more time we would have tied to implement automatic locking when the device goes out of range.