BoardBrights

CS122A: Fall 2018

Hector Barajas

Table of Contents

Introduction

The project I created is a system that controls when and what light patterns appear on the underside of a longboard. The system first determines whether the lights should be on based on a combination of environmental inputs. If the lights are allowed to be on, then they will turn on and show different patterns based off of the speed of the rider. The pattern appears as a wave increasing in brightness/color intensity that scrolls faster as the board is detected to go faster. The colors also change with three different speed levels with slow being <5mph, med being <10mph, and fast being anything over 10mph. There is also an idle light effect for the case that the rider has momentarily stopped moving, but might continue moving. If the rider stops, the idle pattern turns on for 30 seconds where either the rider can continue (thus turning on the regular patterns) or the board is no longer considering to be currently ridden, thus turning the lights off.

The inputs for determining whether the lights should be on start with a photoresistor that detects whether it is dark enough for the lights to be on. Having the lights on in the day time would be a waste of battery power. Next, if the board is tilted, such as if it is being leaned against a wall, the lights are also not allowed to come on. And finally, the lights will only come on if the RPMs of the wheel are high enough. The reason for these three checks are to make sure to conserve the power of the battery, as well as only having the lights turn on only in the correct conditions and not, say, when walking and accidently bumping/spinning the wheel on something.

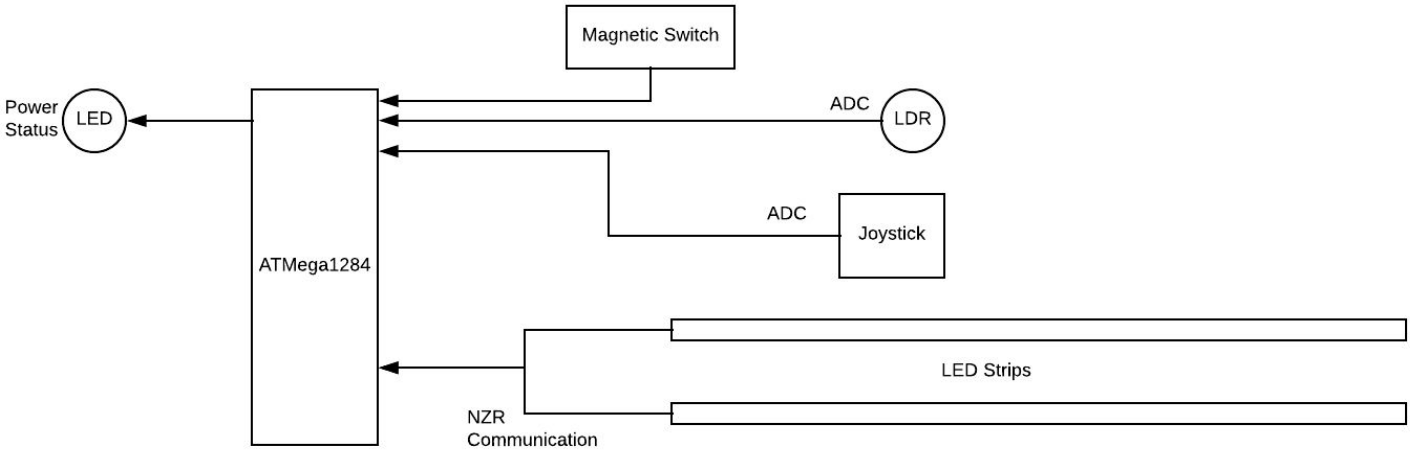
The average RPMs of the wheels is calculated by counting the number of pulses received from a magnet passing by a magnetic switch over the span of 3 seconds. Using the circumference of the 73mm wheels, I calculated that every 6 pulses in three seconds is about 1mph, (i.e. 10 mph would be 60 pulses in three seconds). This way I can control the different speed levels with the average RPMs, and show the corresponding light/color effect.

Hardware

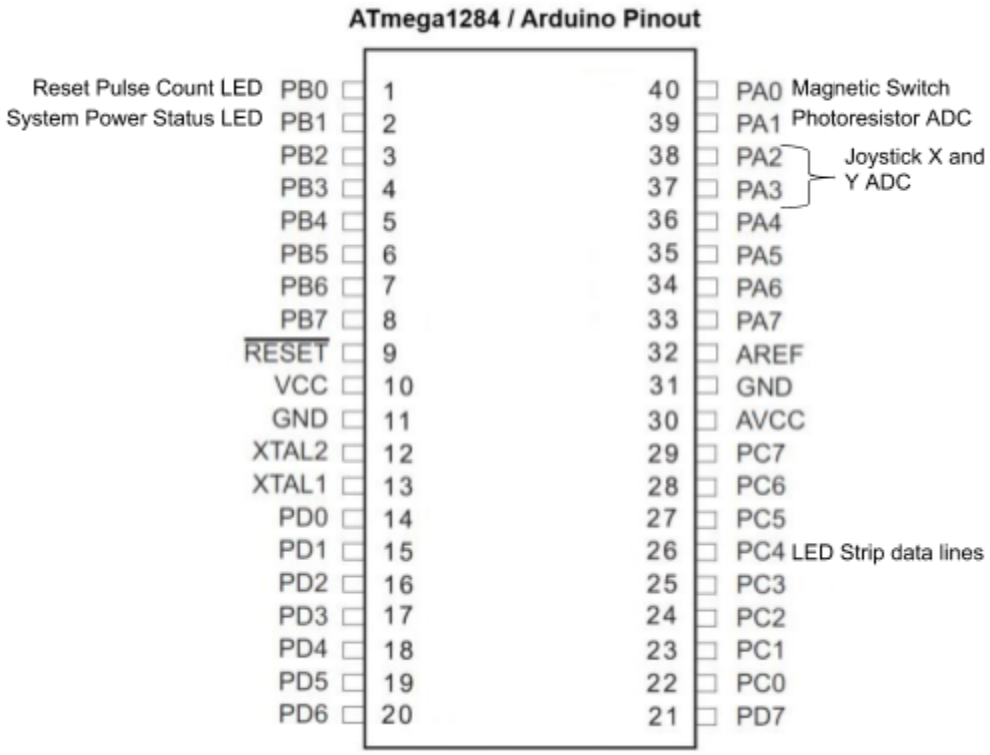
Parts List

Part	Part #	Quantity	Price
ATMega1284	ATMega1284	1	$3.63
joystick	HW-504	1	$5
photoresistor	GM5539	1	$5 /20pc
LED strips	WS2812B	two 2ft strips	$2.25/ft
NC NO Magnetic Switch	NC NO Magnetic switch	1	$5
		Total	$27.63

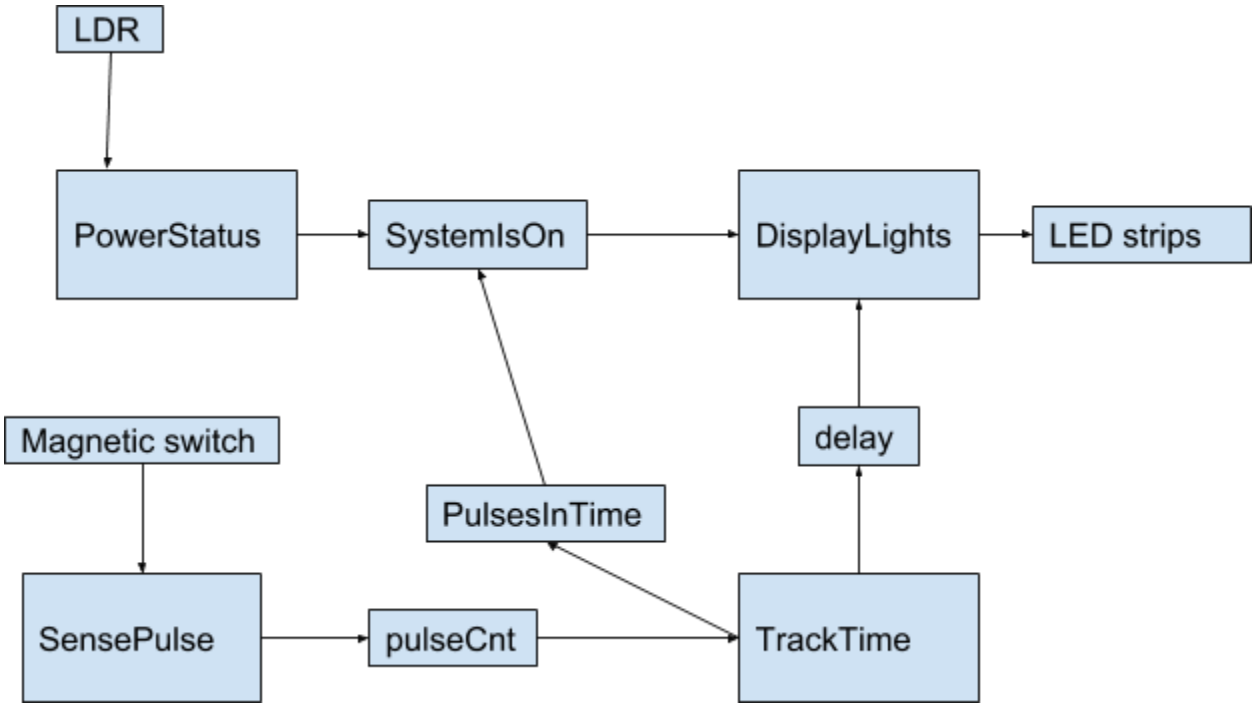
Block Diagram



Microcontroller Pinout



Software



Sense Pulse: Simply detect a rise from the magnetic switch, increment a counter, wait for the fall, and go back to waiting for a rise. Then this writes to pulseCnt which is reset by TrackTime, every 3 seconds in order to grab an average RPM value, PulsesInTime.

DisplayLights: The job of this state machine is to wait a certain amount of time and then scroll to the next phase of the selected pattern. It gets data from PowerStatus through SystemIsOn to determine whether or not the lights should be on. It uses “delay” from TrackTime to know how fast to scroll through the pattern.

PowerStatus: This is more or less the brain of the system which gets information from the environment and decides whether or not the system should be on. It checks the brightness (photoresistor ADC), and was supposed to also check the position of an accelerometer as well as the RPMS of the wheels.

TrackTime: Counts the number of pulses in three seconds and writes to “delay” to tell DisplayLights how quickly to step through the pattern.

Implementation Reflection

The overall project was a great learning experience because it serves a real purpose while also drawing people's attention. I feel that if I were to do this project again, I may consider a different kind of LED strips, and possibly trying to gain more knowledge on the inner workings of the ATMega. The LED strips I chose do not use a form of communication I have heard of. The communication the LED strips use is called non-return-to-zero, or NZR communication, it combines the data line and the clock output into one combined output. For this, I had to find a tutorial on how to interface with the LED strips because the library is made specifically for arduino. The reason I could not use the arduino code is because arduino tend to have external oscillators which I did not know to buy. Therefore, I had to use that tutorial and the code from that tutorial and modify it so that it works with my project. This was on of the biggest setbacks besides the accelerometer issue.

While I am proud of what I built, I am pretty disappointed because I did not get everything that I wanted to work. Specifically, I failed to implement the tilt detection using either the accelerometer+gyroscope, or the joystick. I feel that I could have possibly spend the time to learn I2C on the ATMega or I could have just used an arduino. In any case, I liked the challenge of combining all the systems to detect a rider, brightness levels, and tilt detecting. This is definitely the best school project I have completed and I am the most proud of, and I am glad other people seem to enjoy it as well.

Milestone

My milestones were first to complete the accelerometer, tachometer, and the photoresistor. However, I was held back with the very first component, the accelerometer. That milestone took a long time because I had midterms for the first couple of weeks. Along with that, it turned out that using the accelerometer sensor was much more trouble than I anticipated and I could not get it to work. I didn't find a solution to this issue until the last week, and by then it was too late and I could not get it to work. I feel that using a joystick as an accelerometer was a good idea because the weight that would be attached to the joystick would dampen the vibrations that the actual accelerometer would be subjected to. Unfortunately the noise between using ADC and the NRZ was too high and it failed to work.

Luckily, I was able to get the photoresistor and the tachometer to work appropriately but then I also had trouble with the LED strips. It turned out that trying to interface with the LED strips was not going to be as easy as USART or SPI, instead it used something called NRZ communication which threw me off until I found a blog post on the same issue I had. The code involved some complicated and highly optimized machine code which I could not understand, but the issue it was attempting to solve was the lack of an external oscillator on the ATmega. While the GitHub code turned out to be useful, it was still challenging to get my code integrated with the code which I hardly understood.

Completed components

I completed the most important part of the project which was having light effects on my longboard that change patterns and speed with relation to riding speed. That was my main purpose and it worked.

Incomplete components

The first thing I did not complete is the tilt detection. First I failed with the accelerometer module and then with the joystick. First I didn't know how to use the accelerometer, and then the joystick code seemed to interfere with the LED strip communication interface.

When I was working on the accelerometer, it was especially difficult due to exams, but also because I never used I2C before. I was still in the exploratory phase but I definitely took too much time and I did not want to fall back on my second milestone, which was a more key function to my project. I probably would need a week of work to get I2C to work on the ATMega. But to get the joystick fully implemented, either I might be able to figure out a solution within a few days, or it may be the case that I just will never get it because a lack of understanding of how the Atmega works internally, specifically for the cause of all the noise/interference. Initially the plan was to have the tilt detect somehow use the lights to indicate turning,. Also, the tilt detection could act as a sensor to put the lights in sleep mode. In this case, I believe these ideas aren't too wild and could actually be implemented possibly if I had better components such as an external oscillator and an accelerometer or photoresistor that is not noisy.

Youtube Links

- https://youtu.be/F3VvA3xg9uQ
- https://youtu.be/BgsdTOH_bxE

Testing

LED Strips

The LED strips were easy to test because one the code was transferred, it basically either worked, or it didn't. There wasn't much test here besides changing the output pattern/color/speed and seeing whether the other components caused the right output.

Photoresistor

I tested the photoresistor using an LED bar to see the ADC value at different brightness levels. What I would do is take the value at the brightness I wanted, then calibrate the code with

that value that was output. If the ADC value was, say, 50 when it was dark out, then I would have another LED indicate whether it's passed that threshold.

Magnetic Switch

The magnetic switch did not require much testing. It turned out to be basically a button and using it is similar to using a button. However, to test whether or not it would work properly just mechanically, I had to check whether the period was short enough to detect a highrise and low rise. I used the circumference of the wheel, my top speed (from gps app), as well as the distance from the switch that the magnet has to be, all in order to see how much time the ATMega has to detect a pulse. It turned out to be between 10-15 ms so I kept it at 10ms to be safe.

Joystick/accelerometer

In order to test this I would use a similar strategy as the photoresistor because the output is similar to an ADC value. The LED bar is easy and convenient to use and I basically used it in some way for all of my testing. If the output value passes a certain threshold, then act accordingly. Before hooking it up to the rest of the system, I would test it out with the LED bar and some other indicator LEDs.

Known Bugs

- Glitched LED
    - I believe there is some kind of noise that the ADC operation in the ATMega is somehow interfering with the NRZ communication to the LED strips. For some reason, when the ATMega is grabbing an ADC value, one of the LEDs will glitch, and flash some color. This could potentially be fixed with some kind of timing adjustment, because in the tutorial I read, the author spoke of how critical the timing is.
- System on/off at wrong times
    - I think this also has to do with ADC noise. Basically all of the LED inputs gave me trouble unless it was just one, like the photoresistor. During testing, I put the photoresistor ADC value to the LED bar and I noticed that the value was not at all stable. So as a solution I set a wide range to allow a buffer for the noise, but still, the noise will sometimes initiate the sleep sequence. This is probably a related issue to the one above, or it could also be a case that the components I am using are not of the best quality.

Resume/Curriculum Vitae (CV) Blurb

BoardBrights

BoardBrights is a project I created that utilizes embedded systems within an ATMega1284 microcontroller programmed in C. There are four interacting state machines that determine when is an appropriate time to turn on BoardBrights, and what effects it should display. When on, the speed of the board, as detected by a magnetic switch tachometer, modulates the speed, as well as the color of the light display. Otherwise, a photoresistor ADC value may detect too much brightness and the system goes into a sleep mode to conserve energy.

Future work

The next feature I would add is the tilt detection I have worked so hard on yet failed to implement. I believe that using an arduino for this would be appropriate because of the libraries that are premade specifically for the components I have. Doing so would allow me to focus more and getting more features without worrying about how internals of the hardware work. I'd love to make a case for this project so that I could actually use it day-to-day. Not only would I need to waterproof everything, but I would also need to secure everything in a way the does not fall apart with constant vibration.

I actually believe something related to my project could be used on Boosted boards. LEDs are very visually pleasing and Boosted likely has means by which to handle the vibration, waterproofing, and impact issue that I would have. For this reason I do tend to see my project as something I would present were I to apply to Boosted for a job.

References

Datasheet for LED strips: http://www.nooelec.com/files/WS2811.pdf

Driving the WS2811 at 800 kHz with an 8 MHz AVR: http://rurandom.org/justintime/index.php?title=Driving_the_WS2811_at_800_kHz_with_an_8_MHz_AVR

Github from that author: https://github.com/DannyHavenith/ws2811

Appendix

Some Math:

Wheel radius r = 0.036m, speed cap v = 10mph = 4.5m/s
Circumference of wheel = 2*pi*r = 0.22619m

1 revolution / 0.22619m * 4.5 m / 1 second = ~20 revolutions / second,
or 60 per three seconds as an upper cap for speed (red color show)

20 revolutions / second means 0.05 seconds per revolution. Every tick of the magnet switch takes about ¼ the length of the circumference, therefore, ¼ of 0.05 seconds, or 12.5 ms is the time it takes for the magnetic switch to rise and fall. Therefore, a period of 10ms is chose.

What could have been: Joystick Accelerometer Proposal