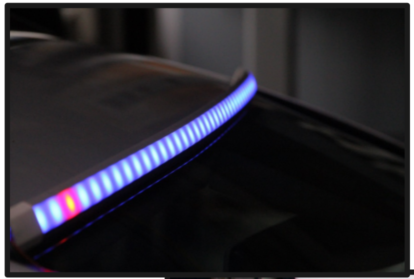
interACT

Designing cooperative interaction of automated vehicles with other road users in mixed traffic environments

Human-machine interface development

interACT: On-board and external HMI

LED strip -- Task

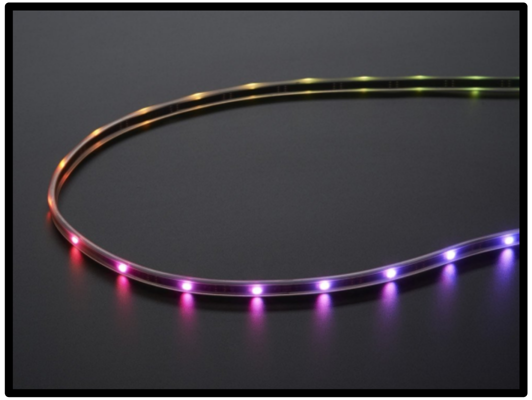


- Program an LED strip with an Arduino
  - LED strip will alert pedestrians of an autonomous vehicle’s awareness to their presence
- Develop signal library for Arduino
  - Write generic code for every LED strip that could be used for interACT projects

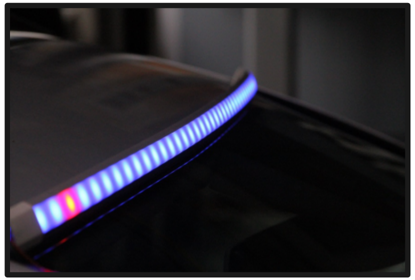
LED strip -- Requirements

Since the LED strip we were using isn't being made anymore, we needed to write a generic signal generator program for the Arduino, not just for the LED strip.

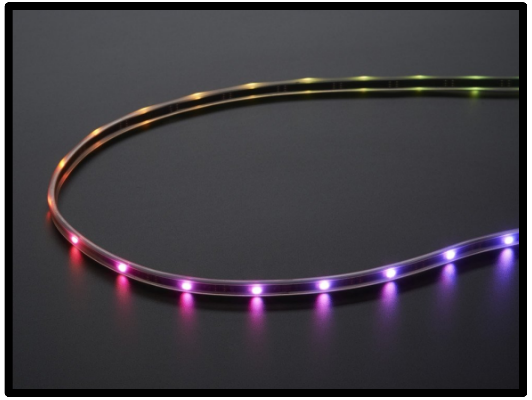
- Have the Arduino output any arbitrary signal on the Arduino's clock cycle
- Work for potentially any LED strip with any data packet length (8-bit or 12-bit)



LED strip -- Issues

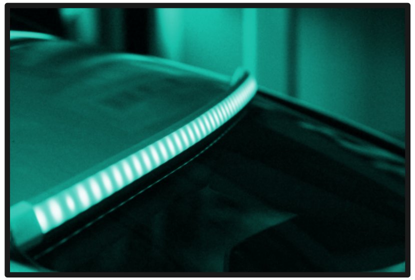


The Arduino has library functions for writing bits to its digital pins, but they are very slow--much slower than the Arduino's clock.

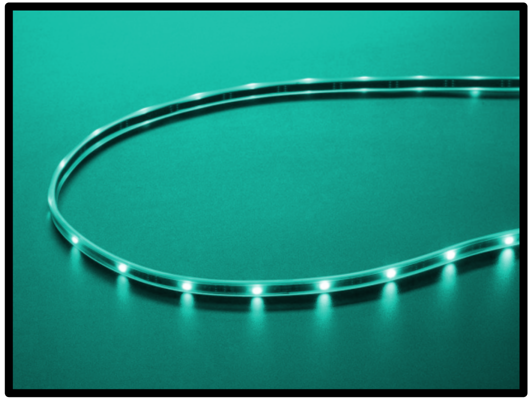


One way to satisfy the LPD1886's stringent timing requirements is by writing Assembly code. But this is difficult to do, and Assembly is not generic for all LED driver ICs.

LED strip -- Solutions



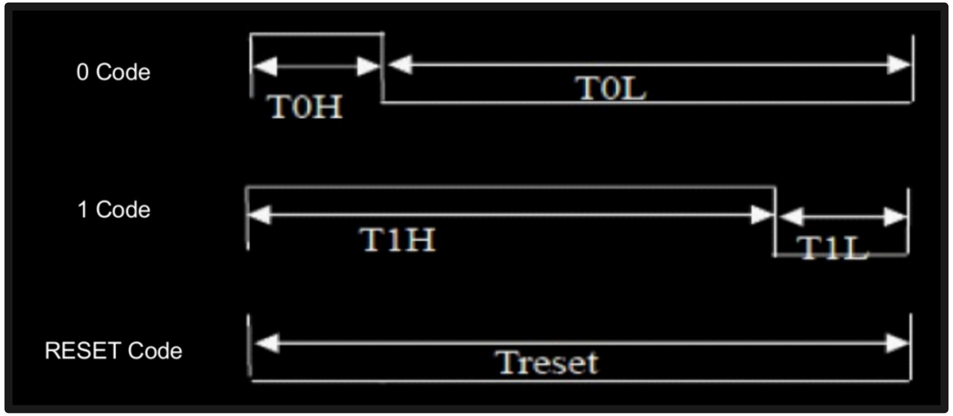
Assembly is the only practical way to program for this LED strip. Therefore, we must create a *generic code writer* to write the Assembly for us.



Our program will accept timing requirements for a given LED strip, and then a C/Assembly code writer will satisfy the timing requirements in-code.

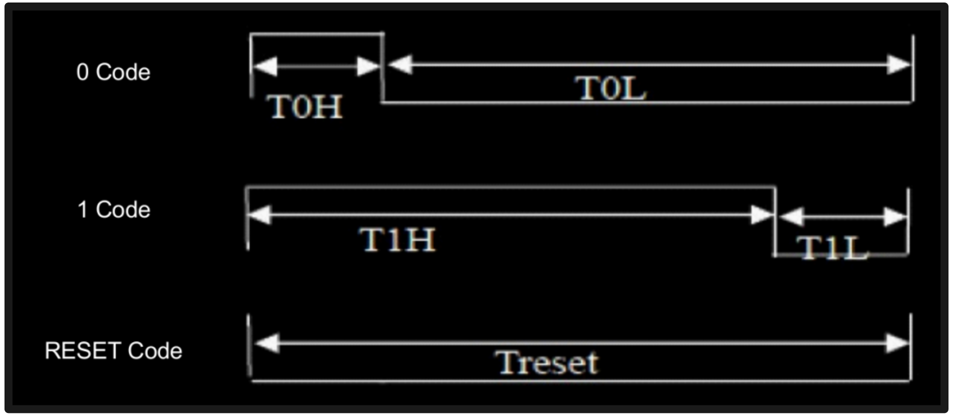
LED strip -- Implementation

- Each ‘HIGH’ and ‘LOW’ bit in the programmer could have as many 0V- or 5V-periods as necessary.
- Each ‘bit’ would be executed on a designated Arduino pin as it arrives from the USB serial input. That is to say the Arduino is essentially a data interpreter and a data transcriber for the LED strip.



LED strip -- Implementation

- The programmer is given the Arduino's timing requirements and clock frequency, and from there constructs correctly-timed Assembly code (or, if the timing requirements are impossible, alerts the user).
- The programmer then compiles and uploads this code to the Arduino.



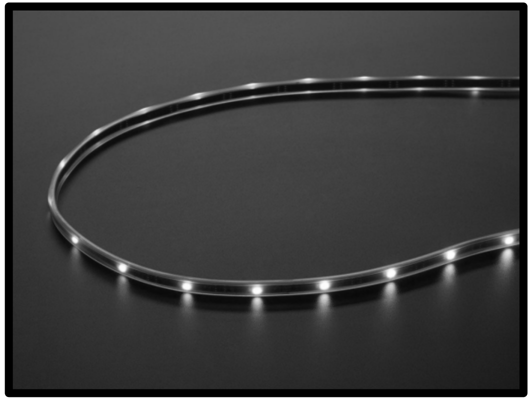
LED strip -- Results

Successes:

- Timing structures were correctly implemented
- Compilation commands were correctly set up

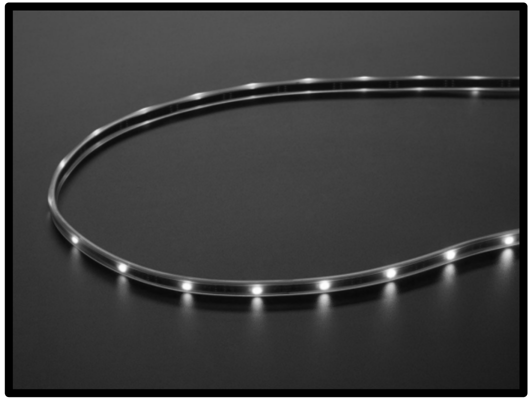
Failures:

- No testing on actual Arduino Assembly was done
  - IT deemed the WinAVR assembler software “unsafe”
  - As a result, the generic Assembly writer was not written
- Code for managing timing structures remained generally untested



LED strip -- Results

Basically what we have now is a bunch of unverified abstract timing structures that *could* accompany an Arduino Assembly writer.



Visual elements -- Task

- Design a standard LED strip animation and pattern data structure
  - To be cross-platform
  - Create a program that can interpret VisualElement_D messages from the Dominion application

Visual elements -- Requirements

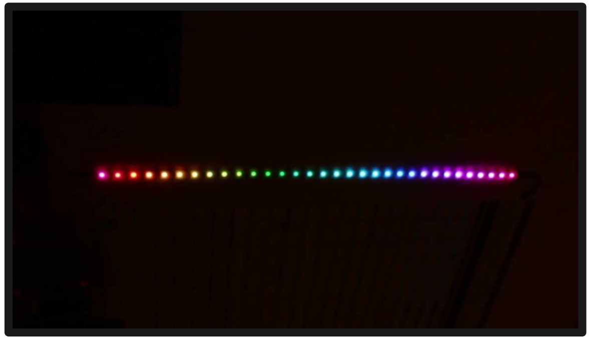
- Way to store the VE representations must be compact and cross-platform
- Reading the VE representation must be relatively simple
- Programs must be able to integrate multiple VisualElement_D’s into LED strip animations
  - VE’s should be ‘stacked’ on one another in an LED strip display, as if they have a z-value.

Visual elements -- Issues

- Sending 12-bit data structures to the Arduino is awkward
  - but necessary because the LPD1886 accepts 12-bit data structures.
  - Data must be formatted before being sent. This requires tedious bit manipulation.

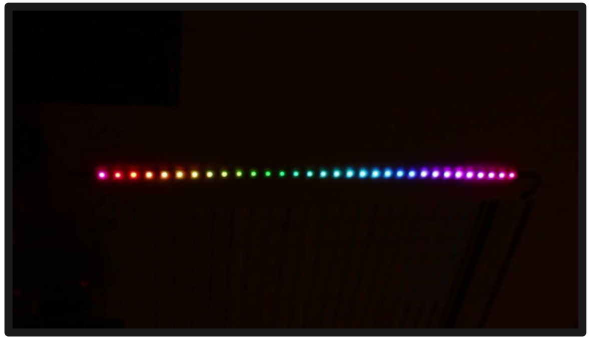
Visual elements -- Implementation

- Create monochrome bitmap files to represent the LED strip being on/off in the pixel locations
  - Architectures can parse the monochrome bitmaps into bool arrays with the help of a lightweight C++ program
- A VisualElement_D struct has a color associated with said bool array, as well as a frame frequency
  - The bitmap (bool array) is cycled through its rows and the color of the VE is displayed onto the LED strip.
- The program keeps an array of VisualElement_D's and 'stacks' them on the LED strip.
- The USB program outputs whatever is necessary to make the programmed pattern appear.
  - Must first manipulate the data into an 8-bit format

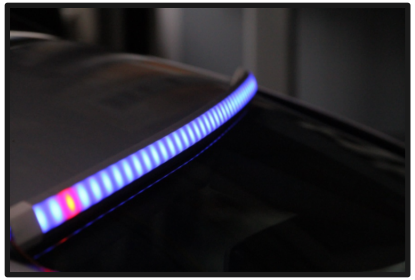


Visual elements -- Results

- Full implementation took too long to make! I plan to send updated versions of the program I wrote later on.
- The Bitmap writer is complete
  - Creates beautiful monochrome bitmaps!
- As of now, the USB serial writer is nearly complete.
  - Testing still needs to be done
- The VisualElement_D manager still needs to be written.



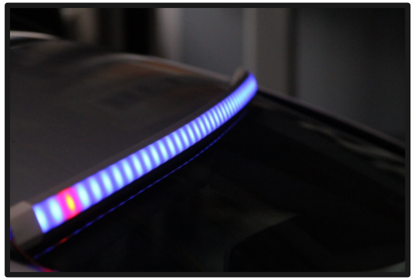
Conclusion



I’ve not only written helpful programs for these applications, but I have (most importantly) written a framework for future work.

With proper progress in each of these projects I’ve been working on, we can expect:

- Interactions between pedestrians and autonomous vehicles though visual structures to be more easily implemented
- DLR and Dominion programming structure to be more organized



Future work

- Obviously, whatever work I didn't finish should be finished
- A GUI for creating LED strip animations should be made to help psychologists and others who may not easily be able to use the bitmap backend
- Other LED strips in different locations on the car can now (soon) be integrated
- Studies can be done to see how individuals on the outside of the car react to the lights on the autonomous vehicle

Thank you!

Questions? Comments? Concerns?

candy?