Display for a Matrix of Pressure Sensitive Fibers

Abhishek Damle

Department of Electrical and Computer Eng.
Virginia Tech

Contact: adamle@vt.edu

Date: April 20, 2020

Table of Contents

I. Overview

The goal of this project was to create a display for an 8x8 matrix of pressure sensitive fibers. A series of voltage dividers were used to interface the pressure sensor with an Arduino microcontroller and a representation of the pressure sensor was displayed on an LCD. Lines were used to represent the individual fibers in the matrix while the intersection of two fibers was represented by a circle. The color of these circles corresponds to the level of pressure applied at the intersection of two fibers and as an increasing amount of pressure is applied, the color of the circle changes from blue to green to red. A block diagram of the system is shown in Figure 1. Since an 8x8 mesh of fibers was not available, the system was prototyped using a 2x3 mesh.

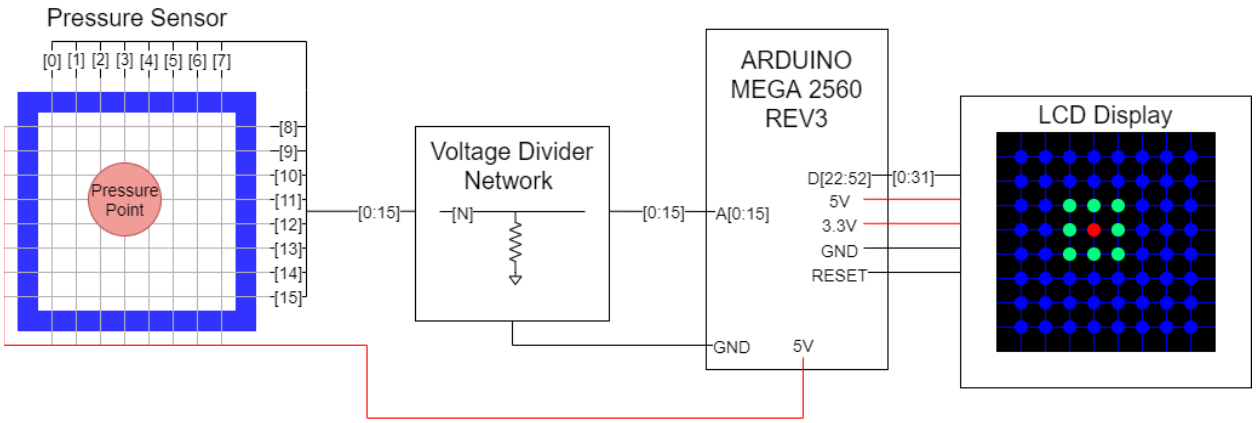

Figure 1: Block diagram of the system.

II. Fiber as a Pressure Sensor

The pressure sensor consists of a mesh of pressure sensitive fibers. These fibers have a carbon-black loaded elastomer core that causes the resistance of the fibers to change as they are strained. The fibers can operate in four different regimes as shown in Figure 2 and the principle behind this phenomenon is described in [1].

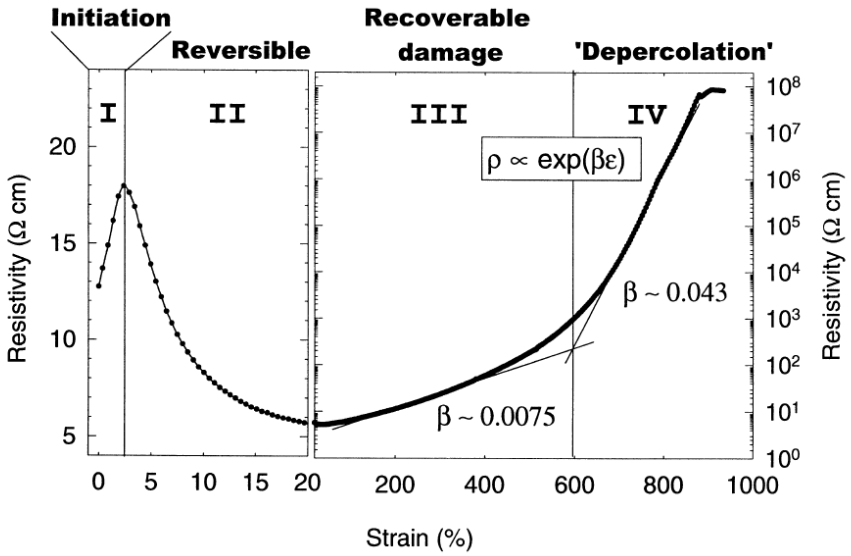

Figure 2: Relationship between the strain applied to a fiber and the fiber's resistivity [1].

The fully reversible nature of regime II allows the fibers to be used as reliable pressure sensors and under this regime the fibers become less resistive as pressure is applied to them. A voltage divider shown in Figure 3 is used to sensor the resistance of the fiber under pressure, and Vout is applied to the ADC of the microcontroller (MCU) for the prototype.

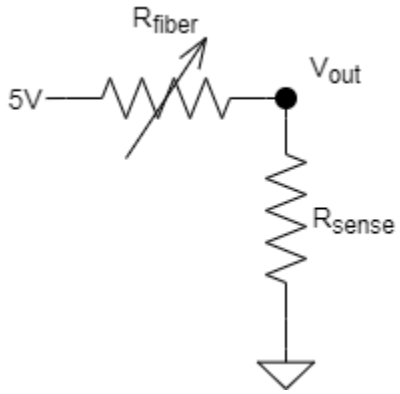

Figure 3: Interfacing circuit with a fiber.

Note R_{sense} should be roughly matched with the resistance of the fiber to ensure that the output of the voltage divider has a large swing. It was observed that the fibers have resistances in the MΩ range, so 1 MΩ resistors were used for R_{sense} for the prototype.

III. MCU

Since the resistance of each of the fibers must be measured independently, a microcontroller with numerous ADC channels was required for this project. Arduino MEGA 2560 REV3 was chosen for this project due to its 16 analog pins and the added benefit of having a simple development environment with numerous libraries. Some of the key properties of the Arduino MEGA 2560 REV3 MCU are listed in table 1.

Table 1: Arduino MEGA 2560 REV3 key properties

Processor	8-bit ATmega2560
Clock frequency	16 MHz
Operating voltage	5 V
Supply voltage	7-12 V
Digital I/O pins	54
# of Analog input pins	16
ADC resolution	10 bits
Power dissipation of processor	100 mW typical

Detailed instructions for installing the Arduino IDE are provided on their website [2] and a tutorial for running a demo program on an Arduino Mega using the Arduino IDE is provided in their guide [3].

IV. Display

An LCD was used for this project as it allows for data from the pressure sensor to be displayed in a dynamic way. The HiLetgo 3.5" IPS TFT LCD was chosen for this project and it installs on the Arduino Mega as a shield. The TFT_HX8357 library found in [4] was used to interface with the display and can easily be installed through the Arduino IDE using the “Add .ZIP library” option. This library is based on the Adafruit GFX library and [5] describes key library functions. Some of the key properties of the HiLetgo 3.5" IPS TFT LCD are listed in table 2.

Table 2: HiLetgo 3.5" IPS TFT LCD key properties

Screen size	3.5”
Screen resolution	480 x 320 pixels
Screen type	TFT
Supply Voltage	5V / 3.3V
Driver IC	HX8357C
Power dissipation	.4 W - .55 W

V. Program Details

The pressure sensor and display are encapsulated by the “pressure_sensor” and “lcd_display” classes, respectively. The “pressure_sensor” class handles measuring the state of the fibers’ while the display is handled by the “lcd_display” class.

A. Pressure Sensor Class

The primary function of the "pressure_sensor" class is to determine and store the state of the pressure sensitive fibers which is the fraction of the current change in the fibers' output to the maximum change in voltage the fibers can experience. The variables are defined as follows.
V_{out} : the sensed voltage of a fiber in Figure 3
V_{max} : the voltage under maximum pressure
V_{min} : the baseline voltage under no pressure

The maximum change of the voltage $\Delta V_{max} = V_{max} - V_{min}$. The expression for the sensed pressure state of each fiber is given in (1).

$$state = \frac{V_{out} - V_{min}}{V_{max} - V_{min}} = \frac{V_{out} - V_{min}}{\Delta V_{max}} \quad (1)$$

The state in (1) denotes the ratio of the voltage change of a fiber under pressure to the maximum possible change ΔV_{max}. The baseline output voltage V_{min} is measured in the beginning of the program execution. ΔV_{max} is measured for each fiber in advance and the value is converted into an integer and stored in a lookup table. This integer value, N, is given by $N = 204.6 \times \Delta V_{max}$. Table 3 shows the ΔV_{max} and converted integer values for each fiber.

Table 3: ΔV_{max} values used in the program

Fiber Index	ΔV_{max} (V)	Converted Integer Value
0	0.176	36
1	0.073	15
2	0.073	15
8	0.976	200
9	1.006	206

The output voltage V_{out} is sampled at 1 ms interval by the ADC, and the average value of ten ADC readings is used to calculate the state from (1). This calculated state value is then rounded to the five discrete values of 0, .5, .65, .80, and 1 to reduce the effects of the noise and minor variations of the fiber. Table 4 shows how a calculated state is converted to a discrete value.

Table 4: Discrete state values corresponding to calculated state values

Calculated state	Discrete state
0 - .35	0
.35 - .5	.5
.5 - .65	.65
.65 - .8	.8
.8 - 1	1

Each fiber is indexed based on which of the 16 analog input pins it is connected to on the Arduino Mega. This index value is referred to as the "fiber_index" throughout the program and is used as inputs for methods of the "pressure_sensor" and "lcd_display" classes. In the program, the

“measure(int fiber_index)” method is used to update the state of a fiber following the previously described procedure. Meanwhile, the “get_percent(int fiber_index)” method can be used to retrieve the state of a fiber.

B. LCD Display Class

The “lcd_display” class displays a model of the pressure sensor, where vertical and horizontal lines represent the fibers while the intersection between the fibers are represented by circles. As pressure is applied to two intersecting fibers, the color of the circle representing the intersection point changes from blue to green to red. Specifically, the color of a circle is determined by its state which is equal to the average values of the states of the two intersecting fibers.

The API for the “lcd_display” class is simple. The display can be updated using the “update_display(int fiber_index, double percent)” method, where “fiber_index” corresponds to the analog port a fiber is connected to and “percent” is the state of the fiber. These two classes make up the bulk of the system and the main loop simply consists of a call to the “measure” method of the “pressure_sensor” class followed by a call to the “update_display” method of the “lcd_display” class.

C. RGB Color Values and Circles

The circles are drawn using the “fillCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color)” function from the Adafruit GFX library where “x0” is the horizontal location of the center of the circle, “y0” is the vertical location of the center of the circle, “r” is the radius of the circle, and “color” is the color of the area of the circle. The “color” variable is a 16 bit value that determines the intensity of the red, blue, and green subpixels. The format for the color variable is described in the Adafruit GFX library[5] and is shown in figure 4.

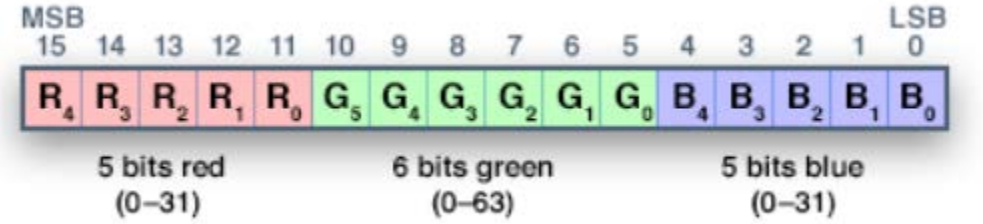

Figure 4: RGB format [5]

As a circle transitions from blue to green to red, the corresponding RGB value of the circle changes from 0000000000011111 to 0000011111100000 to 1111100000000000 in binary. Therefore, there are 188 discrete values between blue and red in this RGB format. The “get_rgb_color” method from the “lcd_display” class essentially takes a percent value as an input and multiplies it by 188 to determine which discrete RGB color corresponds with the given percent input. Since the state of a circle is representative of the ratio of the pressure applied at the intersection to the maximum pressure that can be applied, it can be used as the input to the “get_rgb_color” function to determine the RGB value of the circle. This results in the circles on the display changing from blue to green to red as more pressure is applied to the intersections the circles represent.

C. Flow Chart

A high-level flowchart of the program is shown in Figure 5. The program first measures V_{min} of each fiber without pressure. These baseline voltages make up the V_{max} values from (1) and are used to determine the state of the fibers. The program then enters its main loop. It measures output voltage V_{out} of each fiber, calculates its average, and then its state using (1). Next, the state of each intersection point between a pair of fibers is obtained as the average of the two associated states. Lastly, the updated states of the intersections are converted to color values and are used to redraw the circles.

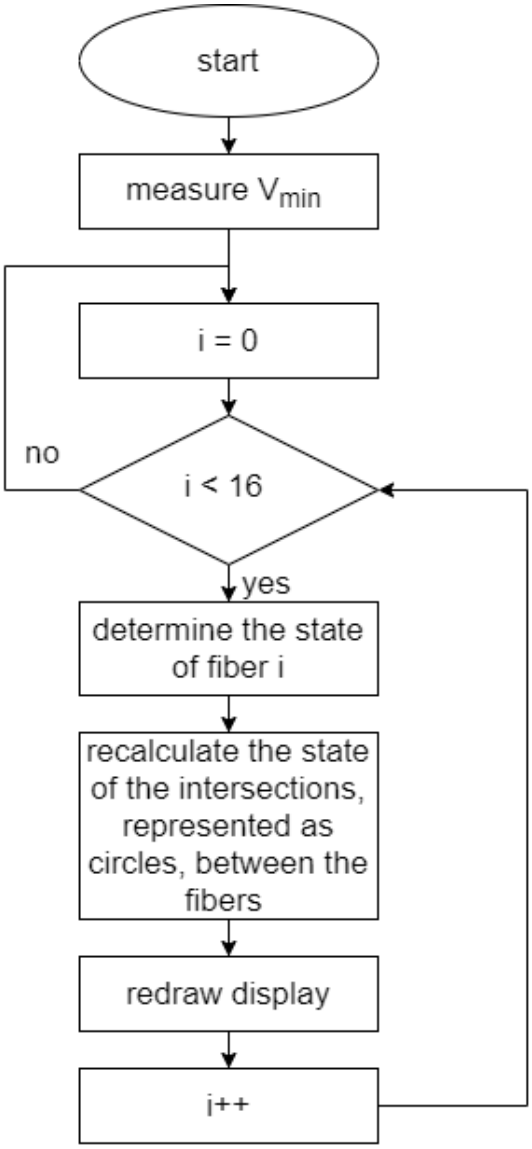

Figure 5: High level flowchart of program.

In addition, debugging code that outputs the state of the fibers through UART is included in the main loop and may be removed for the final version of the program.

VI. Source Code

The source code has been uploaded to the git repository in [6]. The "pressure_sensor" class described in the previous section is defined in the "pressure_sensor.cpp" file while the "lcd_display" class is defined in the "display.cpp" file. The "pressure_sensor_display.ino" file contains the main loop. The member variables, and functions of these two classes along with their purpose is listed in table 5.

Table 5: Member variables and functions in the source code and their purpose

Member variable/function	Purpose
Class: pressure_sensor	
Variables	
int voltage[16]	Store voltages of analog pins
int base_voltage[16]	Store V_{min} values for each fiber
int max_voltage[16]	Store ΔV_{max} values for each fiber
double percent_of_max_voltage[16]	Store state of each fiber
Functions	
void measure(int fiber_index)	Calculate and store state of fiber with index of fiber_index
void reset()	Reset variables and remeasure V_{min} values
int get_voltage(int fiber_index)	Return voltage of analog pin with index of fiber_index
double get_percent(int fiber_index)	Return state of fiber with index of fiber_index
void calibrate()	Remeasure V_{min} values
Class: lcd_display	
Variables	
double fibers_percent[16]	Store state of each fiber
unsigned int lines[16]	Store color of each line in the display
unsigned int circles[8][8]	Store color of each circle in the display
Functions	
void update_display(int fiber_index, double percent);	Update state of fiber with index of fiber_index to percent
void reset()	Reset variables and the display
unsigned int get_rgb_color(double percent);	Return a fiber's state value, percent, to an RGB value
void update_circles();	Update color of circles and redraw
void update_fibers()	Update color of lines and redraw

VII. Part List

Part	Description	Note
MCU	Arduino MEGA 2560 REV3	Reference [7]
LCD Display	HiLetgo 3.5" IPS TFT LCD	Reference [8]
R_{sense}	1 MΩ, Tolerance = 5%	

VIII. Prototype

The system was tested using a 2x3 mesh of pressure sensitive fibers. Figure 6 shows the prototype working correctly for a pressure point applied to the top left of the pressure sensor.

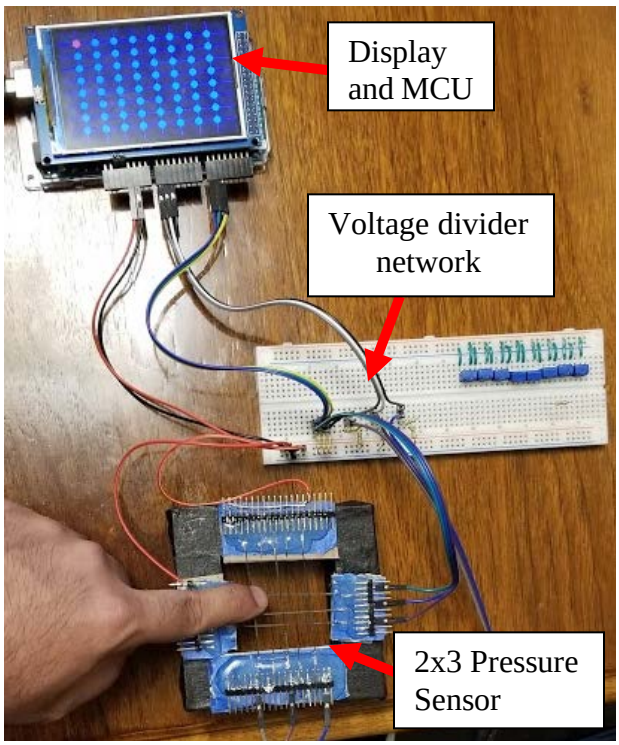

Figure. 6: Picture of the prototype

IX. Future Work

The system was prototyped successfully using a 2x3 pressure sensor. The current program was written to accommodate an 8x8 pressure sensor and requires minimal modification to work with the larger sensor. Specifically, line 21 in “pressure_sensor_display.ino” should be removed and “max_voltage” values in lines 6 to 23 of “pressure_sensor.cpp” need to be updated with the maximum change in voltage for each of the fibers.

X. References

[1] L. Flandin, A. Hiltner, and E. Baer, “Interrelationships between electrical and mechanical properties of a carbon black-filled ethylene–octene elastomer,” *Polymer*, vol 42, pp. 827-838, January 2001

[2] Arduino, “Getting Started with Arduino products,” Dec. 7, 2019. [Online]. Available: https://www.arduino.cc/en/Guide/HomePage. [Accessed: Apr. 8, 2020].

[3] Arduino, “Getting Started with Arduino MEGA2560,” Jan. 1, 2017. [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoMega2560. [Accessed: Apr. 8, 2020].

[4] Bodmer, “Arduino library for HX8357 TFT display,” Sep. 9, 2018. [Online]. Available: https://github.com/Bodmer/TFT_HX8357. [Accessed: Apr. 8, 2020].

[5] P. Burgess, “Adafruit GFX Graphics Library,” Jul. 30, 2019. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/adafruit-gfx-graphics-library.pdf. [Accessed: Apr. 8, 2020].

[6] A. Damle, “display for pressure sensor matrix,” Apr. 8, 2020. [Online]. Available: https://github.com/a-damle/pressure_sensor_display. [Accessed: Apr. 8, 2020].

[7] Arduino, “ARDUINO MEGA 2560 REV3,” [Online]. Available: https://store.arduino.cc/usa/mega-2560-r3. [Accessed: Apr. 13, 2020].

[8] HiLetgo, “HiLetgo 3.5" IPS TFT LCD Display for Arduino Mega2560,” [Online]. Available: https://www.amazon.com/HiLetgo-Display-ILI9481-480X320-Mega2560/dp/B073R7Q8FF. [Accessed: Apr. 13, 2020].

XI. Appendix I: Circuit Diagram

Please refer to the “schematic_pressure_sensor.pdf” file for a schematic of the project.