Light Navigator

A simple Arduino project using a 5-position switch sensor and an 8x8 LED display.

Ryan Hardy
Marisa Smith
Nicholas Warren

Table of Contents

Introduction

(Author: Marisa Smith)

Using an 8x8 LED display (MAX 7219) and a 5-position switch sensor (basically a simple joystick) controlled by an Arduino, we plan to create a simple game. Ultimately, we would like to use the joystick to direct the player character, or pip, represented by a blinking LED around a maze displayed on the 8x8 board. The pip will be able to move in 4 directions. There are a few steps along the way:

1. get the 8x8 display working [done]
2. create a level [done]
3. be able to move the pip in the level [done]
4. use input from joystick [done]
5. failure states [done]
6. win screen
7. more levels

Depending on how well this project is scoped and how hard each step is, we might have to scale back.

The Microcontroller Platform

(Author: Ryan Hardy)

Arduino is an open source electronics platform aimed at providing easy-to-use hardware and software to beginner programmers, artists, hobbyists, and anybody who wants to learn about how software and hardware work together. Anybody can buy one for ~$20-60 depending on add-ons. The Uno is running with a ATmega328P microcontroller. It's a low-power CMOS 8 but with AVR assembly-enhanced RISC architecture. It has throughputs ranges at about 1 MIPS PER MHz. Also it has 32 x 8 general purpose working registers. The part we really played with was the digital and analog ports on the top of the board.

The Test Device

(Author: Nicholas Warren)

MAX7219 Red Dot Matrix Module

This Matrix is an 8x8 LED module. It has 5 pins: CLK, DIN, CS, VCC, and GND. They can be daisy chained in a row to create a display of multiple modules. The VCC and GND are for supplying power to the device. CLK is for the microcontroller to send its clock signal to the devices. DIN is for sending in commands to the devices and CS is for setting the device to receive input.

27801 5-Position Switch

The 5-Position Switch is a little joystick like device that sends a low signal from the position that is selected. The 5 positions are up, down, left, right, and a center switch. So this device does not take any input, but justs sends 5 high input signals. Once one of the switches are selected, it sends a low signal. So the microcontroller needs to monitor these inputs.

Development tools

(Author: Nicholas Warren)

The LEDControl Library was very useful for setting up our LED Matrix. It sets up an LedControl object that has various functionality. We, however, only used 4 methods. Three of these methods were only used for setup and the majority of the work was done with setLed. This function allowed us to specify the exact LED we wanted to work with of the 64 available.

The LedControl object would assign the pins passed to be output and create an array of bytes used to set the rows of up to 8 separate devices. Each time an LED is set you are actually setting the whole row. It then, does a display test to see if all LEDs are functional, sends a decode message to see if the Matrix is functioning properly, clears all LEDs, and finally goes into shutdown.

The function setLed turns on or off the LED at a row by column. The parameters are addr, an int designating the device you wanted to use(always 0 for us), the row of the LED, the column of the LED and a boolean for on or off. So, setLed creates a byte in which zeros represent off LEDs and ones represent on LEDs. This byte is then sent to shiftOut.

In turn called shiftOut, an arduino.h function, to send data directly to the LED Matrix. To call this function the CS pin needs to be set to low, telling the device to get ready to receive data. Then, call shiftOut with the parameters of dataPin (DIN), clkPin (CLK), bitOrder (most bit first) and the value(a byte). First you must send in the opcode, which is the number of the row you want to manipulate, then you send in the byte that sets the LEDs in the row. Once, you are done calling shiftOut, you must turn the clkPin to High so the device knows to stop reading.

To set up the switch all we had to do is set each of the pins that are receiving signals to input mode, this is done with the pinMode function. Once we want to read form these pins we just call digitalRead with that pin number and it returns the value read from the pin.

The Experiment

(Author: Marisa Smith)

We were required to use at least one sensor in this project, but it seemed more fun and interesting to have two affect each other. So, our original idea was to use the 8x8 display and joystick combined. The obvious choice for these components was a simple game; we envisioned a lit dot moving around on the display, controlled by the user.

As we continued discussing, we thought the next step could be to make a maze for the player to navigate around to an “exit,” which would be the win condition. We also discussed that touching a wall would reset the position of the dot, to add some challenge and a failure condition.

To set up the 8x8 LED display, we used these connections (Arduino - Display):

12 - DIN
10 - CS
11 - CLK
5V - DCC
GND - GND

The 5-position switch sensor was set up as follows (Arduino - Switch):

9 - UP
8 - LT
7 - RT
6 - DN
3.3V - VCC
GND - GND

Conclusions

(Author: Marisa Smith and Ryan Hardy)

One of the challenges of this project was debugging. Because the code ran on an Arduino, there was no “easy” way to check what was in a variable through a print statement. The 8x8 board was easier to get working because a lot of people have played with something similar and written about it, and we could instantly see if the part was wired to the Arduino properly, because all of its lights lit up.

For the switch, in the beginning, we didn’t know if the switch was broken or just wired incorrectly. It was also a less popular choice for projects so there was no template we could look at as a guide that was known to work. The switch had no status light on it to indicate that it was on or off. We coded up some status lights on the LED board and used it to debug the joystick by unplugging the cables to the Arduino and seeing what turned on or off. Common sense told us that the joystick would require an analog connection to the Arduino but it actually needed digital. Once we fixed the bugs in hardware, coding it was easy.

Other ideas that have stemmed from this project include:

- an Arduino board that fetches a weather report (somehow?) and displays a corresponding graphic on the 8x8 display
- adding teleports to the game
- finding a board that has multicolored LEDs and using the different colors to represent different things in the game
- maps that change over time
- adding sound with a third device
- a joystick controlled robot with a laser

Contributions

Joystick Controls - Ryan & Nicholas
Project Manager - Marisa
Level Designer - Marisa
Documentation Formatting - Marisa
Game Code - Marisa & Nicholas
Wiring - Nicholas
Requirements - Roie Black

Project code

-LightNavigator.ino
-LedMSControl.h
-LedMSControl.cpp
-arduino.h
-arduino.cpp

Sources

https://www.parallax.com/sites/default/files/downloads/27801-5-Position-Switch-v1.1.pdf

https://www.maximintegrated.com/en/products/power/display-power-control/MAX7219.html

http://www.instructables.com/id/LED-Matrix-with-Arduino/

https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftout/

https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/

https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/

http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf