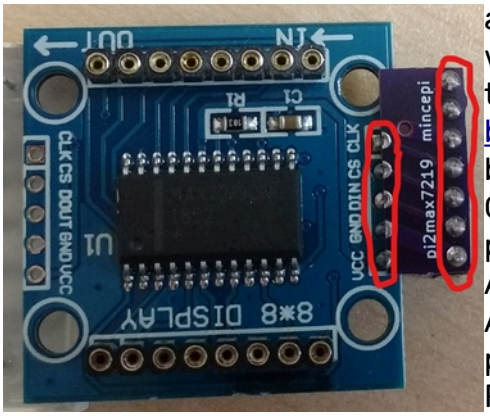
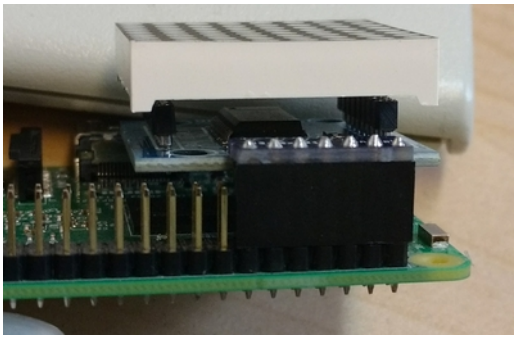
Seattle Raspberry Jam
A Raspberry Pi Meetup for Beginners to Experts
RASPBERRY JAM

Hardware Project #7: LED 8x8 Array

Assembly

In this project, we will be creating a scrolling clock from an 8x8 LED array (chip number MAX7219). We will need an LED display (such as this one on Aliexpress: bit.ly/2YYHu1p), a custom circuit board (you can order this from OSHPark: bit.ly/3072d0d), and a 7-pin socket (such as this one on Aliexpress: bit.ly/2KKX0oN). Assemble and solder these components together as shown in the picture to the left. The final step is to plug the pin socket into the Raspberry Pi's pin header on pins 2 to 14 as shown in the picture to the right. **WARNING! Incorrectly installing this display could damage the Raspberry Pi. Please have one of the organizers check that it is plugged in correctly before applying power.**





The New Way of Accessing GPIOs

The GPIO Character Device

The old method of accessing the GPIO pins through the file system (reading/writing files in the /sys/class/gpio directory) is deprecated as of Linux kernel version 4.8 (used in late Raspbian Jessie). It is being replaced by the GPIO character device, or gpiod. Many benefits come from using this new system, such as automatic pin cleanup if a program using GPIO pins crashes and drastically increased speed. However, while using this system is easy in C programs due to the supplied C library, implementing GPIO in shell scripts is less easy than the former system.

Software Setup

There are many ways of controlling these chips, from Python scripts to Arduino code. For this project, we decided to use C code to create both a file device and a clock. We will be using the GPIO character device C library in our program (see the box above for more information on this utility). Use the Advanced Package Tool (apt) to install it:

```
sudo apt install libgpiod-dev
```

Once that completes, continue on to the next section, where we will download and compile the code.

Continued overleaf

Downloading the Code

First, download the main code from bit.ly/2ZVET5G and the clock code from bit.ly/2OSe9lC. If you do not have access to a web browser, see the box to the right titled "How to Download Files from the Command Line."

Tip: How to Download Files with the Command Line

While there are many ways to download files in the command line, `wget` (derived from World Wide Web get) is one of the simplest. To use this application, type `wget URL` where URL is the web address of the file that you are trying to retrieve.

Compiling the Code

The next step is to compile the code. This can be accomplished with the following commands:

```
gcc -lgpiod -o pi2max7219 pi2max7219.c
gcc -o clock clock.c
```

The first line compiles the main program (*pi2max7219.c*) using `gcc` (GNU C Compiler) into an executable called *pi2max7219* and links it to the *gpiod* library mentioned earlier. The second line does exactly the same for the clock program, except that it does not need linking to the *gpiod* library.

Final Step: Running the Code

Finally, we can run the code! Enter the following command to setup the file device:

```
sudo nice -n -20 pi2max7219
```

This will run the executable file we created earlier, which will spawn a process to monitor the file device, */dev/pi2max7219* and write subsequent commands issued to it to the LED chip. Next, enter this command to start the clock program:

```
./clock
```

And the current time should scroll across the display. Press CTRL-C when you want to exit the program. You may notice that this is not the correct time; that is because the Raspberry Pi is set by default to use GMT (Greenwich Mean Time). To change this to PDT (Pacific Daylight Time), enter `sudo raspi-config`, then (using the arrow and Enter keys to navigate) select "Localisation Option" > "Change Timezone," and follow the on-screen instructions. As long as you are connected to the Internet, the next time you run the clock program, the time should be correct. Congratulations! You have completed the LED 8x8 Clock tutorial! If you feel like a challenge, see the box below. We have also supplied a code reference so that you can implement this display into your own projects.

Code Reference for the pi2max7219 Program

We have added a reference below so that you can implement our driver into your own programs. Firstly, the code creates a file device at */dev/pi2max7219*. The pattern in which the LEDs need to be written to is explained by the command below (which uses `echo` to write data to the *pi2max7219* device):

```
echo "1 3 4 7 10 1F 40 7F" > /dev/pi2max7219
```

This may look confusing, but basically we are writing eight one byte numbers to the device (for the eight pixels in columns, and the eight pixels [bits] in the rows). So instead of thinking of these as base-10 numbers, think of them as base-16, or hexidecimal, numbers. So the first number, since 1_{hex} is 1_{bin}, lights pixel[1,1] (the pixel at row 1, column 1). The second number, 3_{hex}, is 11_{bin}, so it lights both pixel[1,2] and pixel[2,2] (remember that, since this is the second number in the `echo` command, it refers to the second column). The third number is 100_{bin}, so that turns on pixel[3,3]. The fourth, 7_{hex}, is 1111_{bin}, so it light pixel[1,4], pixel[2,4], pixel[3,4] and pixel[4,4]. This continues on for the other four numbers.

Challenge: Make the pi2max7219 and clock programs run automatically on boot in the background so that you do not need to run them each time you login. Hint: the script */etc/rc.local* runs on boot. Do not forget to use absolute paths!