Gertboard

Buffers, LEDs, Buttons, PWM Motor Driver, Open-Collector Drivers,
A/D & D/A Converters, and ATmega328P Microprocessor

The Gertboard is a circuit board that plugs into the first 26 of the Pi GPIO header. It was created by engineer Gert van Loo in 2012 and provides an excellent introduction to how computers can control electronic devices. Once you understand how to use all of its components, you will be able to assemble components that specifically meet your needs. In November, 2013, Gert van Loo brought out another board called the GertDuino with some features that are similar. I have not used a GertDuino, but I expect that it is an attractive alternative to the Gertboard. In these notes and in the workshop, I will describe how to exercise the various features of the Gertboard.

It is essential to download and read the Gertboard User's Manual, Revision 2.0:

http://www.element14.com/community/servlet/JiveServlet/downloadBody/51727-102-1-265829/Gertboard_UM_with_python.pdf

and you should also have the schematics from

http://www.element14.com/community/servlet/JiveServlet/downloadBody/52867-102-1-267216/Assembled%20Gertboard%20Schematics.pdf

Connecting the Gertboard Jumpers

For the demonstration programs, the Gertboard must be set up as follows:

1. **Turn off all power** (`sudo halt` and after 15 seconds when it stops blinking, unplug the Pi power.

2. **Disconnect the Gertboard from the Pi** by unplugging its 26-pin connector.

3. **Remove all jumper wires and pair jumpers (the things that connect adjacent pins together).**

4. **Add a pair jumper to allow Raspberry Pi 3.3 V power to supply the Gertboard 3.3 V circuits:**
   Use a pair jumper to connect the 3.3 V power on header J24 of the Gertboard as shown in Figure 7 on page 10 of the Gertboard User's Guide.

5. **Connect the Pi SPI programming pins to the ATmega328P Processor SPI programming pins:**
   Connect pins GP7, GP9, GP10 and GP11 of the J2 header to the pins of the J23 header as shown in Figure 27 on page 43 of Gertboard User's Guide, but **where the Guide has a connection to GP8, you need to make it to GP7 instead.** This allows SPI programming of the ATmega328P processor using chip select 1 (GPIO7).

   The corrections here were required because of a slight difference between the `spi_bcm2708` driver used in Wheezy Raspbian and the `spi_bcm2835` driver used in the newer Jessie Raspbian system. It was also necessary to modify programs like `Voltmeter.py`, `SPITesting.py`, and `Logging/spiReading.py` to get data from the ATmega328P using GPIO8 instead of GPIO7.

6. **Setup output for `BlinkingLEDUsingTimer0.asm` program to go to LED D1**:
   Connect pin PD6 of J25 to pin B1 of J3 and attach a pair jumper to B1-out next to U3. These allow the program to blink LED D1.

7. **Setup output for `BlinkingLEDUsingTimer1.asm` program to go to LED D4**:
   Connect pin PB1 of J25 to pin B4 of J3 and attach a pair jumper to B4-out next to U3. These allow the program to blink LED D4.

8. **Setup output for `BlinkingLEDUsingTimer2.asm` program to go to LED D6**:

Connect pin PB3 of J25 to pin B6 of J3 and attach a pair jumper to B6-out next to U4. These allow the program to blink LED D6.

9. Connect a voltage divider to supply a test voltage for the `Voltmeter.asm` program:

Connect the yellow wire at the center of a 1500 Ω/1000 Ω voltage divider to pin PC1 of the Gertboard

Connect the free end of the 1500 Ω resistor of that divider to a ground on the Gertboard.

Connect the free end of the 1000 Ω resistor of that divider to a 3.3 V terminal on the Gertboard.

Connect pin PB2 to GP7 on header J1. This allows the Pi to ask for data from the ATmega328P.

Connect pin PD3 of J25 to pin B2 of J3 and attach a pair jumper to B2-out next to U3. The program will then makes LED D2 blink when the Pi is getting a voltage reading.

10. Connect a 12 VDC motor to the motor controller:

Connect the + from 12 V batteries to MOT+ of J19 on the Gertboard.

Connect the – from 12 V batteries to MOT- of J19 on the Gertboard.

Connect non-stepping 12 VDC motor to MOTA and MOTB of J19 on the Gertboard (order unimportant).

Connect GP17 on the J2 header to MOTB of J5 on the Gertboard.

Connect GP18 on the J2 header to MOTA of J5 on the Gertboard.

11. Connect a 12 VDC fan to the open collector relay 1:

The instructions in the Gertboard User's Guide on pages 24-27 apply to this test.

Connect GP4 on the J2 header to RLY1 on the J4 header.

Connect as shown in Figure 16 of page 26 in the Gertboard User's Guide. Note: All RPWR pins on J9 are equivalent. These have a maximum of 50 V and 500 mA. If you need more voltage or current, use these to activate a power relay.

12. Connect the stepping motor drivers and stepping motor:

Connect Stepping motor yellow and blue to pins 10 and 15, respectively, of Allegro A4973 #1.

Connect Stepping motor red and green to pins 10 and 15, respectively, of Allegro A4973 #2.

Connect pin GP22 of Gertboard header J2 to pin 8 or Allegro A4973 #1.

Connect pin GP23 of Gertboard header J2 to pin 7 or Allegro A4973 #1.

Connect pin GP24 of Gertboard header J2 to pin 8 or Allegro A4973 #2.

Connect pin GP25 of Gertboard header J2 to pin 7 or Allegro A4973 #2.

Connect the logic+ of the Allegros to 3.3 V on the Gertboard.

Connect the logic– (ground) of the Allegros to a ground pin on the Gertboard.

13. Double-check all your wiring. A mistake will usually just cause it to not work, but occasionally can destroy a component!

14. Plug the Gertboard onto the GPIO pins of the Raspberry Pi. Be sure it is aligned with the connector ends at the SD card end of the Pi becoming flush. There should be 7 pairs of pins left unconnected at the USB end of the Pi.

15. Apply power to the Pi (and therefore also to the Gertboard through the GPIO connector).

Programming the ATmega328P Microprocessor

The Raspberry Pi can compile assembly code, send it to the Gertboard ATmega328P microprocessor, and receive data acquired. This is an excellent way for students to become familiar with assembly language programming of microprocessors.

650 pages of documentation for the ATmega328P are freely available by Googling “ATmega328P complete datasheet” and choosing the top entry “[pdf] Datasheet”. I was unable to find it directly from Atmel, the company that makes the processor; they only offered a 39-page summary.

Pages 1-298 of the complete documentation contain the essential information for programming. A very useful

table of registers is given on pages 612-614, a table of instructions is given on pages 615-618, and finally, the Table of Contents is on pages 642-647.

Assembly programs can be written using the `vim` editor which has a convenient color-coding algorithm that matches most assembly language instructions. The `avra` assembler is then used to convert it into an "object" program. The object program is uploaded from the Pi to the ATmega328P on the Gertboard by a Python program called "`uploadingViaSPI.py`". Finally, the processor starts executing the program as soon as the upload is complete.

To upload object programs to the ATmega328P, we will use the SPI interface of the Raspberry Pi. It must be first enabled by doing

```
sudo raspi-config
```

Selecting choice 8 "Advanced Options", then selecting choice A6 "SPI", enabling it and having it start upon boot up. It will then be necessary to reboot the Pi:

```
sudo reboot
```

Now, we do the following to examine the program, assemble and upload it, and see a blinking LED:

```
cd ~/Programming/Assembly/BlinkingLED/
vi BlinkingLEDUsingTimer0.asm
```

Look it over and notice that it expects the file `m328Pdef.inc` to be in the next directory above the current directory. Quit the editor and compile `BlinkingLEDUsingTimer0.asm` by doing:

```
avra BlinkingLEDUsingTimer0.asm
```

This will produce the file `BlinkingLEDUsingTimer0.obj` which now must be sent to the Atmega328P on the Gertboard by using a python program named `uploadingViaSPI.py as follows`:

```
sudo ../uploadingViaSPI.py BlinkingLEDUsingTimer0.obj
```

When using the newer Python version, Python3, use uploadingViaSPI-Python3.py instead.
If the wiring on the Gertboard is properly set, the light should then start blinking. The uploading programs needed to use GPIO7 instead of GPIO8 to prevent the spi_bcm2835 driver in the Jessie Raspbian system from resetting the chip during the middle of a reload.

The other `.asm` programs in the `/home/pi/Programming/Assembly` directory and its subdirectories are run in a similar manner except that `Voltmeter.asm` and `LoggingWithSPIInterrupts.asm` (used with a Geiger counter) use auxiliary python programs `VoltmeterTest.py` and `spiReading.py`, respectively, to bring data from the ATmega328P back to the Pi.

Studying their code and playing with them while referring to the huge ATmega329P datasheet is a nice way to get started with assembly language programming.

The program `LoggingWithSPIInterrupts.asm` monitors the clicks of a Geiger counter, assigns times to each click with 1 ms precision, and sends the data (when requested by the Pi) over the SPI interface for storage on the SD card of the Raspberry Pi. It is described at

http://yosemitefoothills.com/Electronics/RaspberryPi/GeigerCounter.html

with the actual ATmega328P assembly program listing is at

http://yosemitefoothills.com/Electronics/RaspberryPi/loggingWithSPIInterrupts.asm

The processor program uses interrupts when:

- the timer overflows
- the Geiger counter sends a click pulse
- the Raspberry Pi has requested data
- all data has finished being sent.

Interrupts allow the processor to run without missing data as long as the interrupt handling routines are not too

long. Use of the microprocessor allow the Raspberry Pi to do other independent tasks concurrently. Trying to use the Raspberry Pi directly to catch the pulses of the Geiger counter would not work well because the Pi is doing many varied tasks at once like networking, displaying data, etc.

Running the C Gertboard Examples

In the directory `~/Programming/C/Gertboard` are some programs written in C that do things in the Gertboard. These are described in the Gertboard User's Guide. For example, the `motor.c` program is compiled and run as follows:

```
cd ~/Programming/C/Gertboard
cc -Wall motor.c -o motor
sudo ./motor
```

Running the Gertboard Examples

The python Gertboard programs are run in a similar manner. For example, `motor-rg.py` is run by doing:

```
cd ~/Programming/Python/NoGraphics/Gertboard/
sudo ./motor-rg.py
```