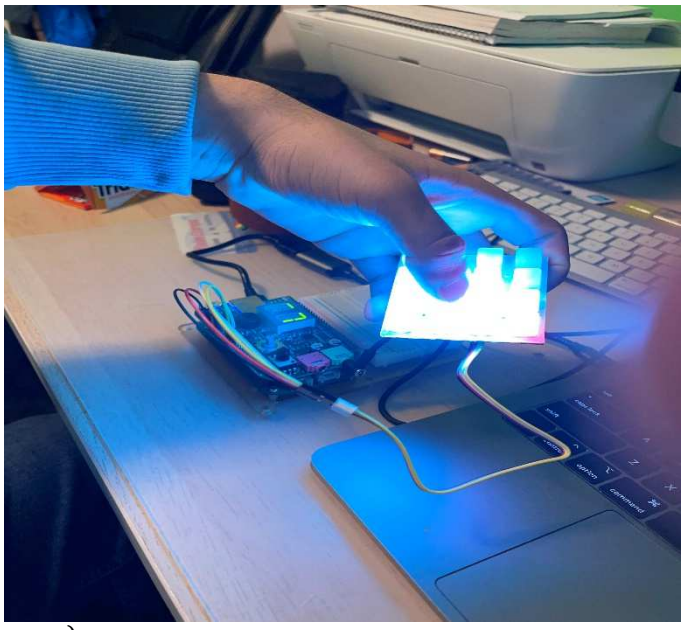
Wiring and control controlling

Adafruit's NeoTrellis 4x4 LED and button matrix



by Audio Group (Spring 2022)

Last update: April 13, 2022

Target device: BeagleBone Green

Target OS: Linux 4.9+

Preamble

This how-to guide expands on the guide provided by previous CMPT433 students with the focus on explaining the Adafruit's SeeSaw chip's register structure and functions. It is better to use this guide in conjunction with the previous students' guide. The link to the guide is provided in the *Reference* section and C file examples are provided in the *Support Files* section.

Table of Contents

1. Wiring the matrix to BeagleBone Green

1.1 You will need the following wires:

- 4 Jumper Male to Female Wires
- 1 Female to Female 4 pin connector cable

1.2 Connecting to the BeagleBone:

The matrix has 4 pins:

- **GND** (ground)
- **VIN** (power)
- **SDA** (data)
- **SCL** (clock)

First, we need to look on the BeagleBone's P8 and P9 expansion headers to determine which pins we need to connect the matrix to.

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

P8

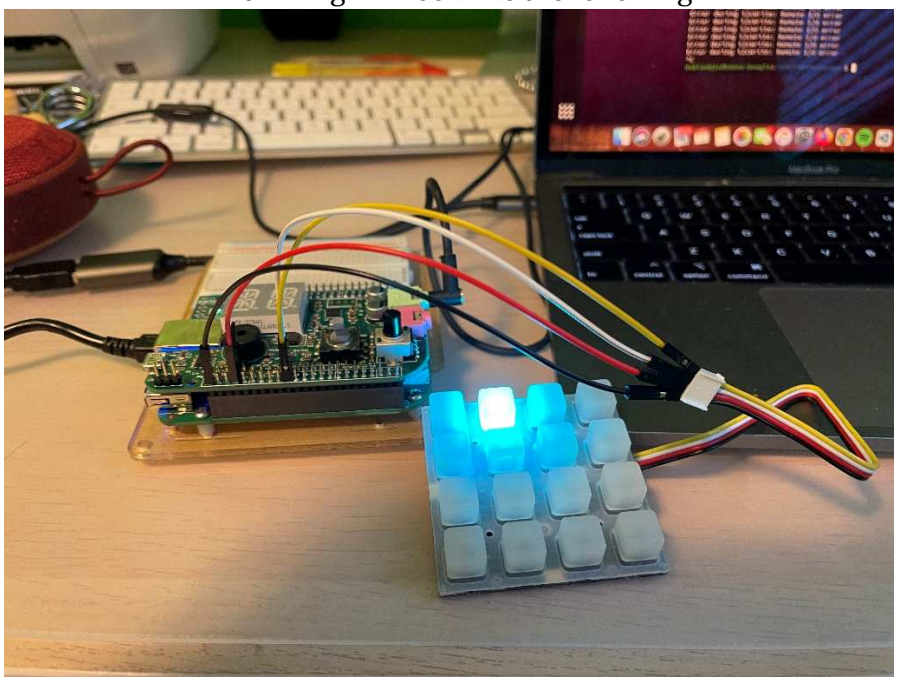
DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

We will connect the matrix to the BeagleBone in the following way:

GND pin to P9_1
VIN pin to P9_7
SCL pin to P9_19
SDA pin to P9_20

This particular wiring connects the matrix to the BeagleBones I2C2 bus, but if you wish to connect to the I2C1 bus, use P9_17 (SCL) and P9_18 (SDA) instead.

The wiring will look like the following



1.3 Check if the wiring was successful

Run the following command on the BeagleBone to display all devices connect on the I2C2 bus:

(bbg)$ i2cdetect -y -r 2

The output should be:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- 2e --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- UU UU UU UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

The matrix has a default address of **0x2e** (46 in decimal), dump its internal registers with the following command:

(bbg)$ i2cdump -y 2 0x2e.

The output should be:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff XX    ...............X
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

2 I2C communication protocol with Adafruit's seesaw chip

2.1 I2c writing to the seesaw chip

To write into the seesaw chip, we need to send 2 **register bytes first** followed by data bytes.

The first byte is the **module base register address** which indicates which module of the matrix we want to communicate with. You can find all the available module registers from Adafruit's website in the *Reference* section.

The second byte indicates the **module function register address** which specifies the desired function within the module.

I2C write transaction will look like the following:

I2C write header	Module base register (1 byte)	Module function register (1 byte)	Data to write

2.2 I2c reading from the seesaw chip

To read from the seesaw chip register, we need to initiate a write to the desired module base and module function registers first. After allowing a short delay, send a standard i2c read header to the chip.

I2C read transaction will look the following:

i)

I2C write header	Module base register (1 byte)	Module function register (1 byte)

ii)

I2C read header	Buffer to read to	Size of the buffer

3 Controlling the matrix's LEDs and Buttons

3.1 Lighting up the LEDs

All the registers that deal with LEDs of the matrix are in the module base register located at **0x0E**, called **NeoPixel**.

Within the NeoPixel module there are 4 module registers that we used:

i) **PIN** – address: 0x01 – 8 bits
ii) **BUF_LENGTH** – address: -0x03 – 16 bits
iii) **BUF** – address: 0x04 – 32 bytes
iv) **SHOW** – address: 0x05 – 0 bytes

PIN register indicates the pin number to output from. We will use 0x01 for the pin register. **BUF_LENGTH** register indicates the size of the internal LED buffer (in bytes). We will use the size of 48 bytes (1 byte for each of the RGB component with 16 pixels in total).

BUF register holds data for RGB components. The first 2 bytes indicate are the offsets for a pixel's index on the matrix.
SHOW register updates the output.

3.2 Controlling Buttons

All the registers that deal with the buttons of the matrix are in the module base register located at **0x10**, called **Keypad**.

Within the Keypad module there are 3 module registers that we used:

i) **KeypadEvent –** address: 0x01
ii) **KeypadEventCount –** address: 0x04
iii) **KeypadFifo –** address: 0x10

KeypadEvent register sets an event for a particular event key. The event key for a button at index x is calculated by the formula*: x/ 4 * 8 + x % 4*. There are three possible events that can be set for a button:

- **TRELLIS_RISING_EDGE,** registers an event when a button is pushed. It has the value of **0x11**.
- **TRELLIS_EVENT_FALLING_EDGE,** registers an event when a button is released. It has the value of **0x09.**
- Both TRELLIS_RISING_EDGE and TRELLIS_EVENT_FALLING_EDGE.

KeypadEventCount register stores the number of events that the matrix has registered since the last time a read from the register has occurred.

KeypadFifo buffer stores events that have occurred in a FIFO manner. To map the registered event **x** to an index in the matrix we need to first apply the following procedure: key = (x >>2) & 0x3F.Then apply the following formula to the result: *key / 8 * 4 + key % 8*.

4. References

i) Previous students' how-to guide:
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/links/files/2019-student-howtos/AdafruitNeoTrellis4x4LedButton.pdf
ii) Overview of the seesaw chip:
https://learn.adafruit.com/adafruit-seesaw-atsamd09-breakout?view=all#using-the-seesaw-platform
iii) I2C guide:
https://opencoursehub.cs.sfu.ca/bfraser/grav-cms/cmpt433/guides/files/I2CGuide.pdf