How-To Guide for 8x8 LED Matrix

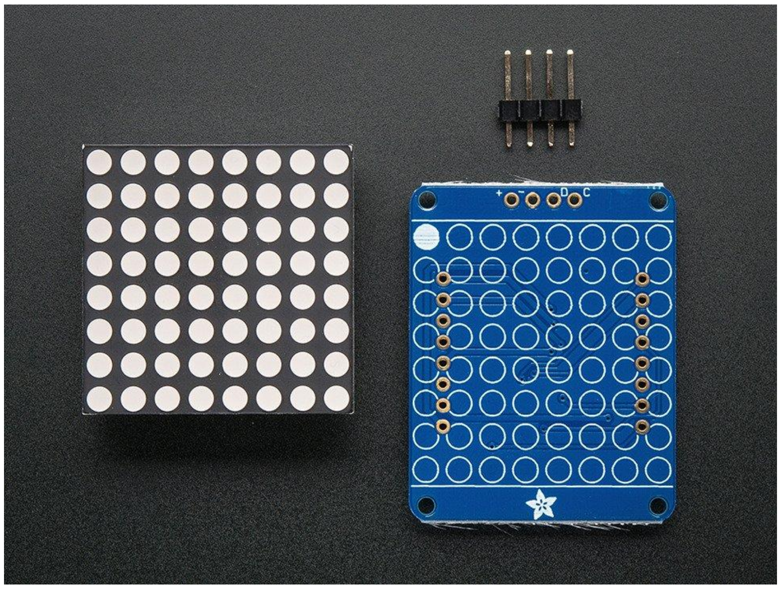
For the Mini-Games Console, a 1.2" square monochrome (green) 8x8 LED matrix board from Adafruit was chosen as the main display. The board has a Holtek HT16K33 LED matrix driver chip which has an I2C interface for controlling the 8x8 LED display. The Adafruit website provides schematics for the board: (https://learn.adafruit.com/adafruit-led-backpack/downloads)

I. Ordering The Board

The Adafruit part number for the board is 1632 and it can be ordered directly from the Adafruit website (https://www.adafruit.com/product/1632) or other resellers like Amazon, Digikey, or Mouser. Because of availability, the 8x8 LED kits were ordered from Elmwood Electronics (https://elmwoodelectronics.ca/products/adafruit-small-1-2-8x8-led-matrix-w-i2c-backpack-red) for C$19.99 plus shipping.

II. Assembling The Board

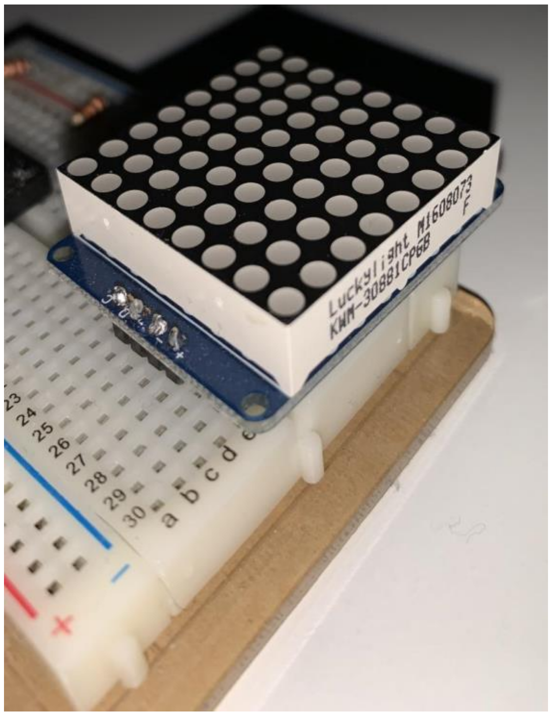
The 8x8 LED kit came with an 8x8 LED matrix, a PCB with the driver chip already soldered, and a 4-pin header for connecting to the BeagleBone Green. The 8x8 LED matrix and 4-pin header have to be manually soldered to the board.
Tools needed: soldering iron and solder.



The Adafruit website provides instructions for soldering the LED matrix and 4-pin header to the board:
https://learn.adafruit.com/adafruit-led-backpack/1-2-8x8-matrix

The web page also shows how to plug the 4-pin header into a breadboard, such as the one provided in the BeagleBoard Green kit for the CMPT433 course, and then solder the 4 pins using the breadboard as mechanical support.

After completing the soldering, the 8x8 LED matrix board plugged into a breadboard should look something like the following:



III. <u>Wiring Connections</u>

The board has markings for the 4 signals of the 4-pin header.
From left to right (see picture above), the signals are:

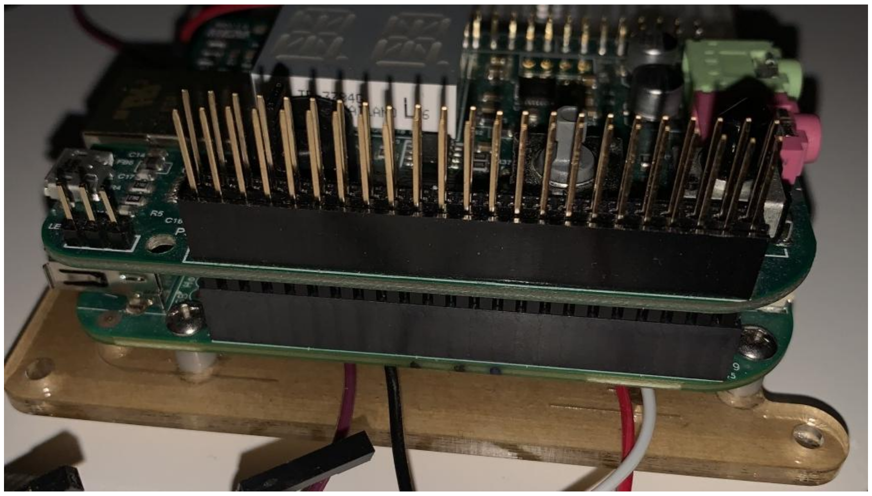
‘C’ = CLK (or SCL) signal for the I2C interface
‘D’ = DAT (or SDA) signal for the I2C interface
‘-’ = GND (ground) for power
‘+’ = VCC+ (positive) for power

Use jumper wires provided with the BeagleBone Zen Cape kit which have a male pin on one end and female connector on the other end to connect the 4 signals between the BeagleBone Green board and the 8x8 LED matrix.

The I2C channel 1 (I2C1) signals of the BeagleBone P9 connector (pins 17 and 18), which is shared with other devices on the Zen Cape board, will be used to connect to the 8x8 LED matrix board.

STEPS:

1) For a more solid connection on the P9 connector, plug in the 2x23 connector with long leads (as shown below) that was included with the Zen Cape kit into the P9 connector on the Zen Cape board.

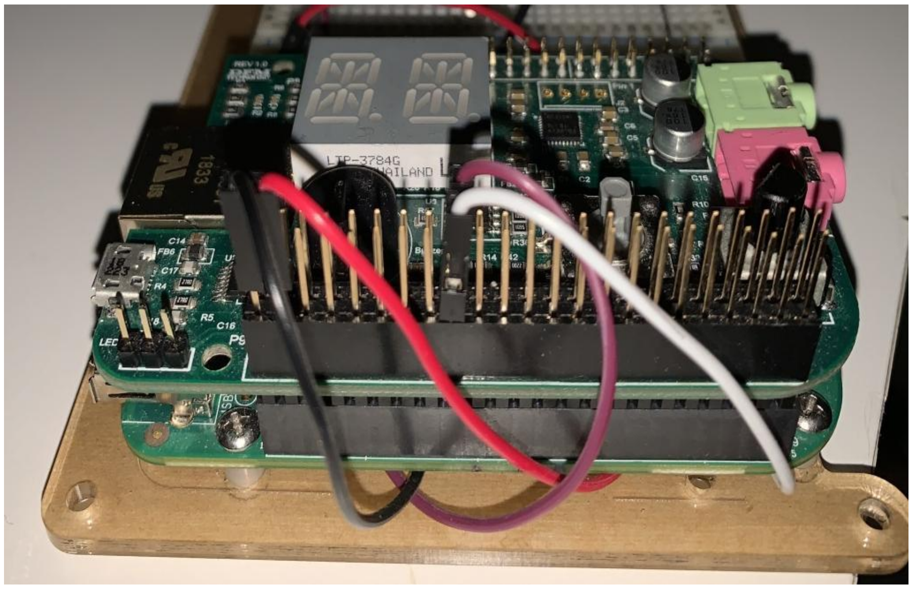


2) Use 4 of the jumper wires mentioned previously and connect the female end to the following pins on the long P9 pins on the Zen Cape:

   P9 pin 1 = GND
   P9 pin 3 = 3V3
   P9 pin 17 = CLK signal for I2C
   P9 pin 18 = DAT signal for I2C

   The row of P9 pins on the edge of the board are the odd numbered pins (1, 3, 5, …, 45) and the row of pins beside it are the even numbered pins (2, 4, 6, …, 46).

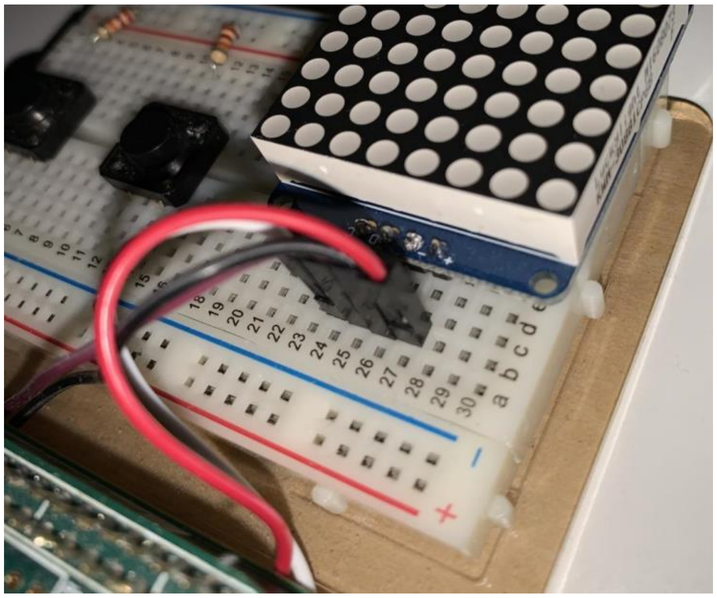
   It is recommended to pass the jumper wires underneath the BeagleBone board from the breadboard area to the P9 side of the board. The following picture shows the wires after making these connections:



3) Connect the male pin end of the 4 jumper wires to the corresponding columns on the breadboard where the 8x8 LED matrix board is plugged into:

   P9 pin 1 wire <==> ‘-’
   P9 pin 3 wire <==> ‘+’
   P9 pin 17 wire <==> ‘C’
   P9 pin 18 wire <==> ‘D’

   The picture below shows the wires after making the connections:



IV. <u>Programming</u>

The Holtek HT16K33 controller's I2C slave address is 0x70. The datasheet can be found here:
https://cdn-shop.adafruit.com/datasheets/ht16K33v110.pdf

Initialization

There are only a few registers to write for initialization:

1) System Setup Register (command code 0x20):
   Write 0x21 (bit 0 of command code = 1) to turn on system oscillator.
2) Dimming Register (command code 0xE0):
   Write 0xE8 (bits 3-0 is 8) to set the brightness level (0 to 15) to middle brightness.
3) Display Setup Register (command code 0x80):
   Write 0x81 (bit 0 is 1) to turn on the display with no blinking (bits 2-1 is 0).
4) Display RAM (command code 0x00 followed by 16 bytes):
   Write 0x00 (command code) followed by 16 zeros to initialize all pixels to 0 (off).

Pixel Locations Mapping

Each pixel is one bit and 8 pixels fit into one byte. The HT16K33 can actually drive up to 16 columns x 8 rows (lines) of LEDs. This is why there are 16 bytes (2 bytes per row) sent for the "Display RAM" command code. For the 8x8 LED matrix, only the lower 8 bits (D7-D0) of each row of the display RAM in the HT16K33 are used.

An 8-byte (1-byte column x 8 row) local display buffer (array) is used to hold the 8x8 pixels. Routines which draw pixels for the display write to this local display buffer. The first byte (array element 0) of the local buffer corresponds to the top line of dots of the 8x8 LED matrix, and the eighth byte (array element 7) corresponds to the bottom line of dots. For each byte, bit 7 corresponds to the left most dot in the row and bit 0 corresponds to the rightmost dot.

A pixel remapping routine is needed to map the local display buffer pixels into a transformed buffer to be sent to the HT16K33's display RAM so that the pixels are displayed correctly at the right locations on the 8x8 LED display. Some experimentation is needed to determine how the pixels are mapped from the HT16K33's display RAM to the 8x8 LED matrix by looking at the schematic of the Adafruit 1632 board to determine the connections between HT16K33 and the 8x8 LED array, and writing some test patterns to the local buffer to determine how pixels show up on the display. The Adafruit website does not have any description of which bit in the display RAM of the HT16K33 controller map to which dot on the 8x8 LED array.

Software Routines Module for 8x8 LED Display - “ext_8x8led.c”
The final static (local) ***extLED8x8RemapIcon()*** function in “**ext_8x8led.c**” is the result of the previously mentioned experimentation and it remaps a local 8x8 buffer into a transformed 8x8 buffer for sending to the HT16K33 controller’s display RAM to correctly display pixels on the 8x8 LED matrix. There is also a public ***extLED8x8SetDisplayRotation()*** function for setting the rotation (0, 90, 180, 270 degrees) between the local buffer image and the displayed image. The rotation setting is factored into the ***extLED8x8RemapIcon()*** function.

The ***extLED8x8Init()*** function performs the initialization described earlier.
The ***extLED8x8DisplayUpdate()*** function remaps the local display buffer to a transformed buffer and sends the transformed buffer to the HT16K33 to display on the 8x8 LED matrix.
The ***extLED8x8DisplayOn()*** function turns on the 8x8 LED matrix.
The ***extLED8x8DisplayOff()*** function turns off the 8x8 LED matrix.
The ***extLED8x8DisplayBrightness()*** function sets the brightness level of the 8x8 LED matrix.

The ***extLED8x8FillPixel()*** function fills the display with either 0s (pixel off) or 1s (pixel on).
The ***extLED8x8DrawPixel()*** draws a pixel (0=off or 1=on) on the local display buffer.
The ***extLED8x8LoadImage()*** loads an 8x8 icon image (8 bytes) on the local display buffer.