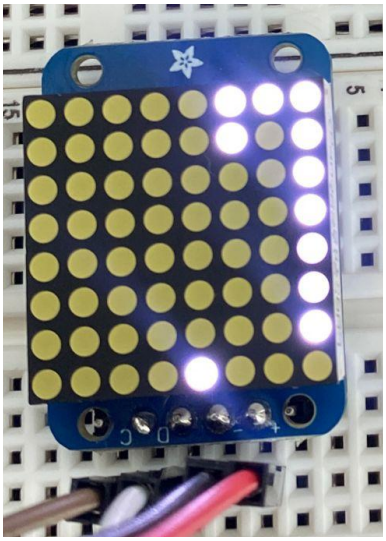
How to implement a 16x16 matrix using 8x8 matrices

By: Yashdeep, Kai, Savinu and Valentina

Introduction:
Implementing a 16x16 matrix using 8x8 matrices on a BeagleBone helps us to optimize hardware resources while expanding display capabilities. This approach not only enhances the visual output but also shows efficient matrix manipulation techniques. In this guide, we'll explore the methodology involved in constructing a 16x16 matrix using 4 sets of 8x8 matrices that communicate with a BeagleBone board through a serial communication protocol called I2C (Inter-Integrated Circuit).

Individual Hardware Setup:
The 8x8 LED matrix contains four different pins that need to be soldered with its backpack (Blue piece under the matrix). To do this follow the instructions in the link: Assembly | Adafruit LED Backpacks | Adafruit Learning System.

You should end up with four soldered matrices as shown in this image:

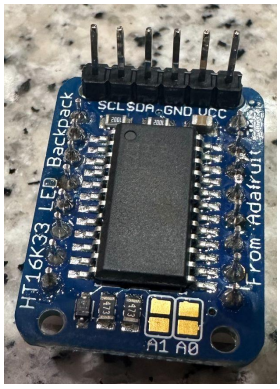


** Make sure to set + - D C pins at the bottom to achieve the correct row positions mentioned in the software part.

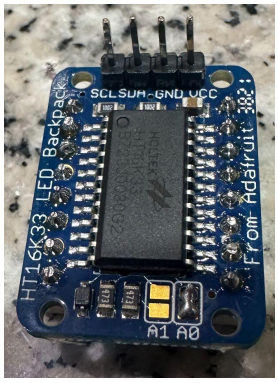
Integrated Hardware Setup:

- Soldering:
  The default device address for the 8x8 LED matrix is 0x70 which does not include any soldering on the A0 and A1 locations.

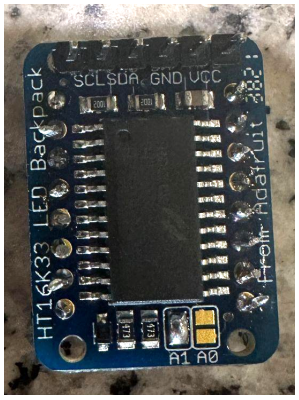
  To be able to write to different matrices, different device addresses are needed. To achieve this we can solder A0 and A1 by making bridges. To get the correct addresses refer to the following images:



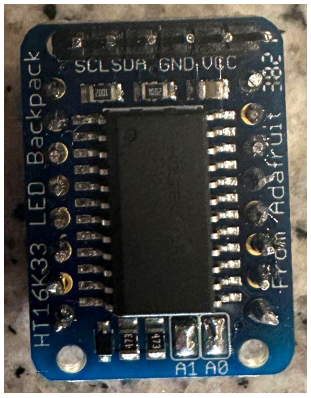
Default: no bridges
Address: 0x70



A0 Bridged
Address: 0x71



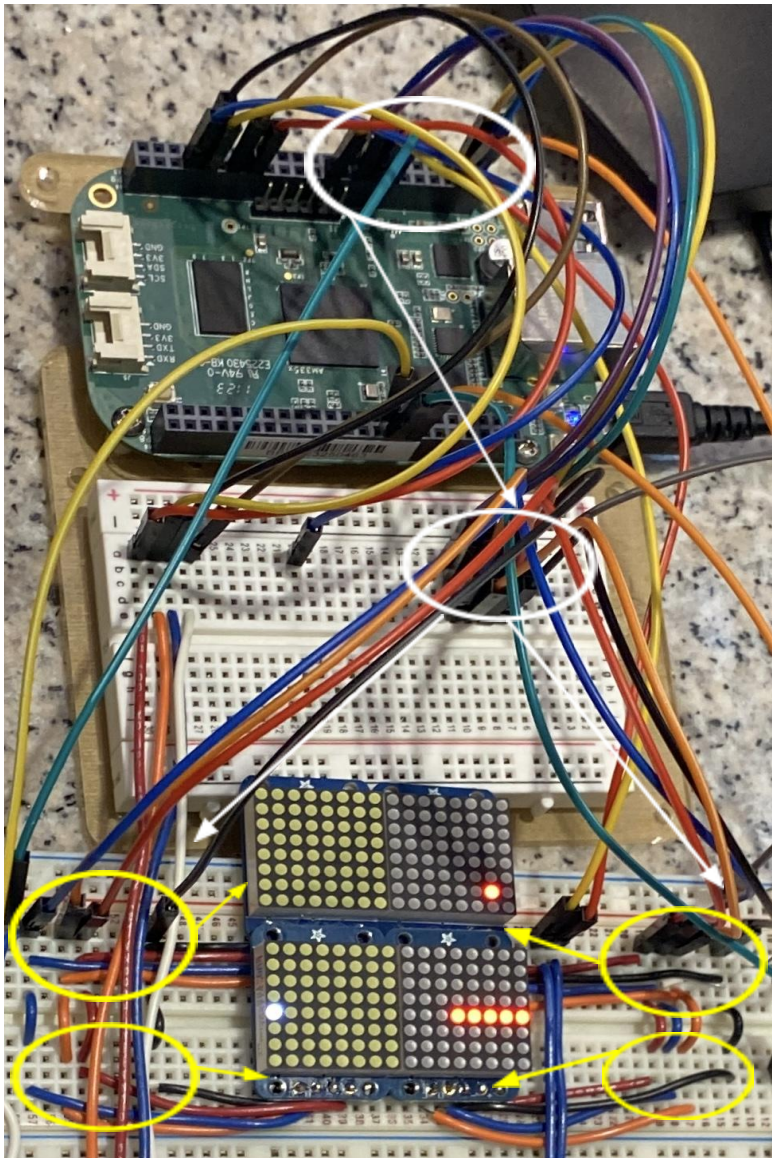
A1 Bridged
Address: 0x72



A0 + A1 Bridged
Address: 0x73

- Wiring:
  Recall that all of the matrices need to have the same pins from the BeagleBone into their corresponding pins.
  1. To have better organization and flexibility on connections, connect the BeagleBone pins to a breadboard in the following order:
     P9.01 (Ground) → to “-” pin on the LED matrix
     P9.03 (3.3V) → to “+” pin on the LED matrix
     P9.17 (I2C1-SCL) → to “C” pin on the LED matrix
     P9.18 (I2C1-SDA) → to “D” pin on the LED matrix

  2. From the breadboard connect the 4 pins that come from the BeagleBone board to the actual +, -, C and D pins on the LED matrix.



Software Setup:
Once the Hardware setup is done all that is left is to implement some functions and a small program in C to make the LEDs light up. Follow the steps in this section to achieve it.

Recall that each row of the matrix is a binary of 8 bits where 1 = ON and 0 = OFF. Therefore, we recommend that you should use an array of eight elements where each element is an 8-bit binary (unsigned char).

1. Configure the two I2C pins (17 and 18) by running the following commands:
config-pin p9_17 i2c
config-pin p9_18 i2c

```
static void runCommand(char* command) {
    // Execute the shell command (output into pipe)
```

```
    FILE *pipe = popen(command, "r");
    // Ignore output of the command; but consume it
    char buffer[1024];
    while (!feof(pipe) && !ferror(pipe)) {
        if (fgets(buffer, sizeof(buffer), pipe) == NULL)
            break;
        }
        // Get the exit code from the pipe; non-zero is an error:
        int exitCode = WEXITSTATUS(pclose(pipe));
        if (exitCode != 0) {
             perror("Unable to execute command:");
             printf(" command: %s\n", command);
             printf(" exit code: %d\n", exitCode);
       }
}
```

2. Initialize the I2C bus with the file descriptor. Notice that this will have to be done with each of the 4 LED matrices i.e. call the function with each of the addresses.

```
int initI2cBus(char* bus, int address) {
    int i2cFileDesc = open(bus, O_RDWR);
    if (i2cFileDesc < 0) {
             printf("I2C DRV: Unable to open bus for read/write (%s)\n", bus);
             perror("Error is:");
             exit(-1);
      }
    int result = ioctl(i2cFileDesc, I2C_SLAVE, address);
    if(result < 0) {
        perror("Unable to set I2C device to slave address.");
             exit(-1);
    }
    return i2cFileDesc;
}
```

Make sure to close each file descriptor at the end of the main function by using `close(i2cFileDesc);`

Call the function:

```
int i2cFileDesc1 = initI2cBus("/dev/i2c-1", 0x71);
```

3. Initialize the I2C LED matrix (display). This will also have to be done for the four devices.

```
//Initialize the display by writing commands
void initI2cDisplay(int i2cFileDesc) {
writeI2cReg(i2cFileDesc, REG_SYSTEM_SETUP, 0x00); //To turn on matrix
writeI2cReg(i2cFileDesc, REG_DISPLAY_SETUP, 0x00);  //To turn on display
}
```

**Use writeI2cReg() function provided in step 4.

Call the function:
```
initI2cDisplay(i2cFileDesc1);
```

4. Finally, write to each of the LED matrices by choosing the correct row and device address using the following function:

```
void writeI2cReg(int i2cFileDesc, unsigned char regAddr, unsigned char value) {
    unsigned char buff[2];
    buff[0] = regAddr;
    buff[1] = value;

    int res = write(i2cFileDesc, buff, 2);
    if (res != 2) {
        perror("Unable to write i2c register");
            exit(-1);
    }
       }
```

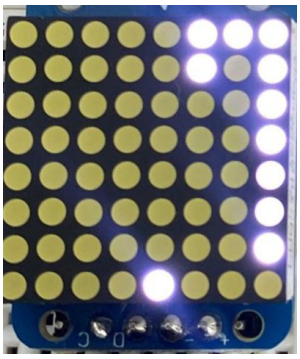
Call the function:

```
writeI2cReg(i2cFileDesc1, 0x0E, 0b00010000);
writeI2cReg(i2cFileDesc2, 0x04, 0b00010101);
```

Software Remarks & Troubleshooting:

➢ Since the + - D C pins are located at the bottom of the LED matrix, here is a diagram of the rows and the corresponding addresses in the right order.

0 0 0 0 0 0 0 1	→ 0x0E	First row
0 0 0 0 0 0 1 0	→ 0x0C	
0 0 0 0 0 1 0 0	→ 0x0A	
0 0 0 0 1 0 0 0	→ 0x08	
0 0 0 1 0 0 0 0	→ 0x06	
0 0 1 0 0 0 0 0	→ 0x04	
0 1 0 0 0 0 0 0	→ 0x02	
1 0 0 0 0 0 0 0	→ 0x00	Last row



Note: Provided binary matrix does not match with the picture

➢ Make sure to initialize all four device addresses: 0x70, 0x71, 0x72 and 0x73. Refer to the previous steps to know what needs to be run for all the matrices.

➢ The LED matrix has a special mapping (physical frame) that differs from what you write in the binary (logical frame).
i.e. 0b00001111 will not directly light up the 4 right-most LEDs. Make sure to identify this mapping

and apply the corresponding binary shiftings as needed.

➢ Check that pins P9.17 and P9.18 are configured to I2C. Refer to the first step.