

## **M14-Exercise 2 for M14-Assignment**

This is MyWire library example for OLED display and Launchpad Board

This dynamically displays alternating patterns on the OLED

## M14- Exercise 2 – MyWire Library & Example Talking Points

- Program layout – uses interrupts differently in a bad way
- Foreground vs Background
- Use interrupt to provide power-down waiting (alternative to polling)
- Create Delay function with Timer to provide power-down delay

File Edit **Sketch** Tools Help

- Verify/Compile Ctrl+R
- Upload Ctrl+U
- Upload Using Programmer Ctrl+Shift+U
- Export compiled Binary Ctrl+Alt+S
- Show Sketch Folder Ctrl+K
- Include Library**
- Add File...

```
178  
179  
180  
181  
182  
183  
184  
185  
186  
187 default: break;  
188 }  
189  
190 }  
191  
192 //This code NOW runs with my MSP430FR2433 +  
193 //After the code is executed, the display s  
194 //(0xA5 -> all pixels on).  
195 // https://e2e.ti.com/support/microcontroll  
196 //  
197
```

Manage Libraries...

**Add .ZIP Library...**

Contributed libraries

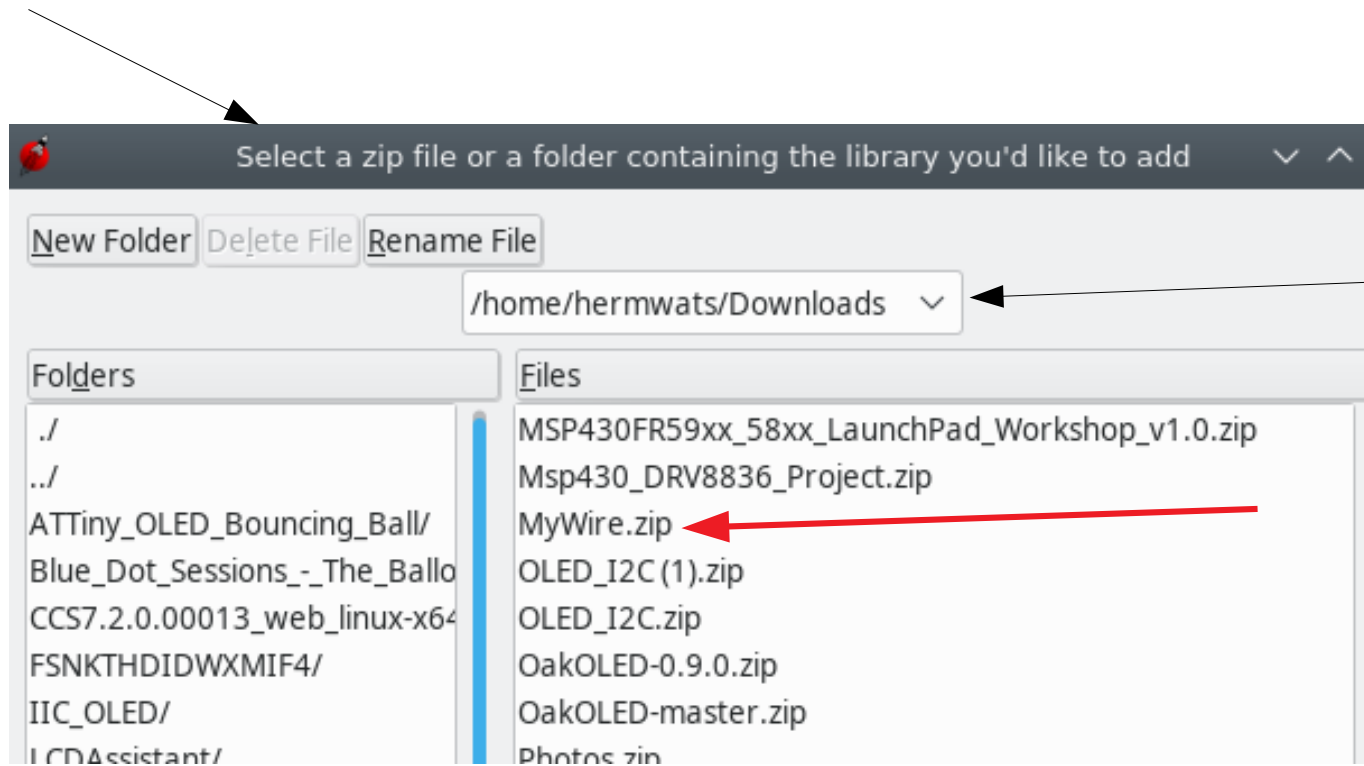
- ACROBOTIC SSD1306
- Adafruit TMP006
- Adafruit TMP007 Library
- Adafruit\_GFX
- Adafruit\_SSD1306
- CogLCD
- GOFi2cOLED-master
- Grove - OLED Display 0.96
- LCD\_SharpBoosterPack\_SPI
- M2XStreamClient

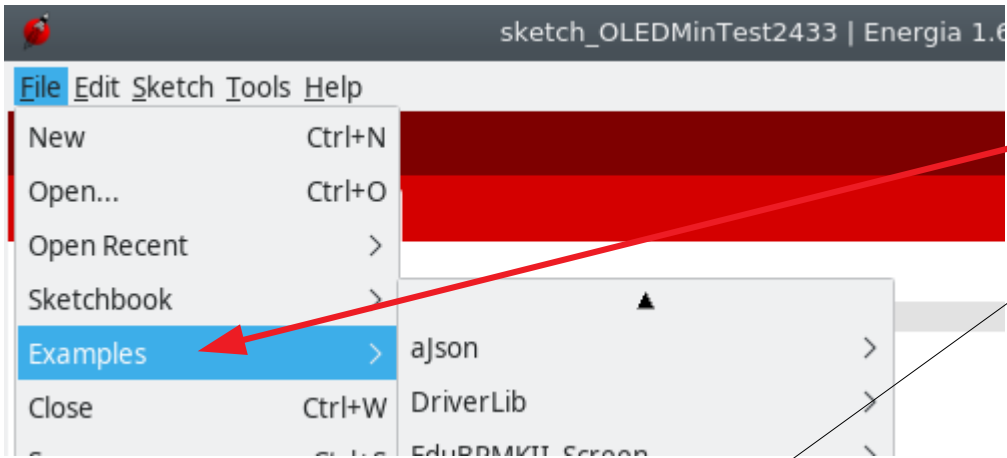
Download the MyWire.zip library from the web site  
Open Energia and Add as a ZIP library

Done uploading.

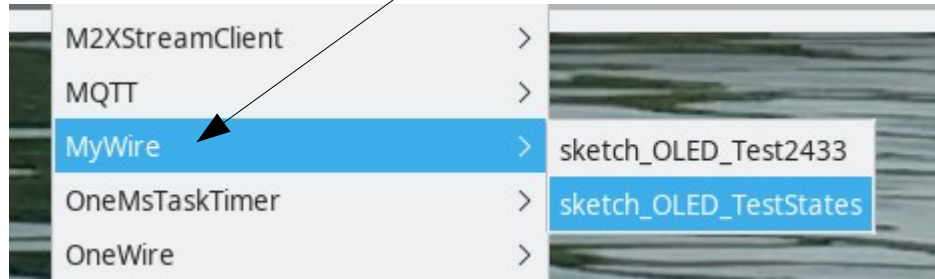
Setting PC to entry point.: 73%  
info: MSP430: Flash/FRAM usage is 488 bytes. R  
Running...  
Success

Continued next slide

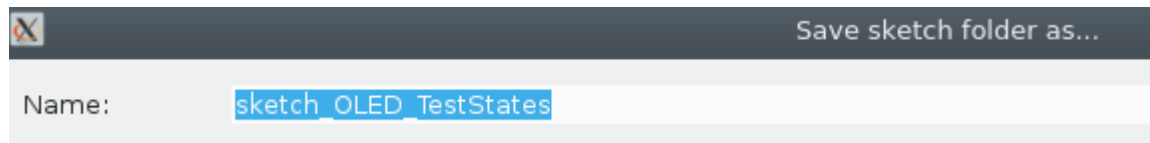




Find sketch\_OLED\_TestStates Under Examples for the MyWire Library

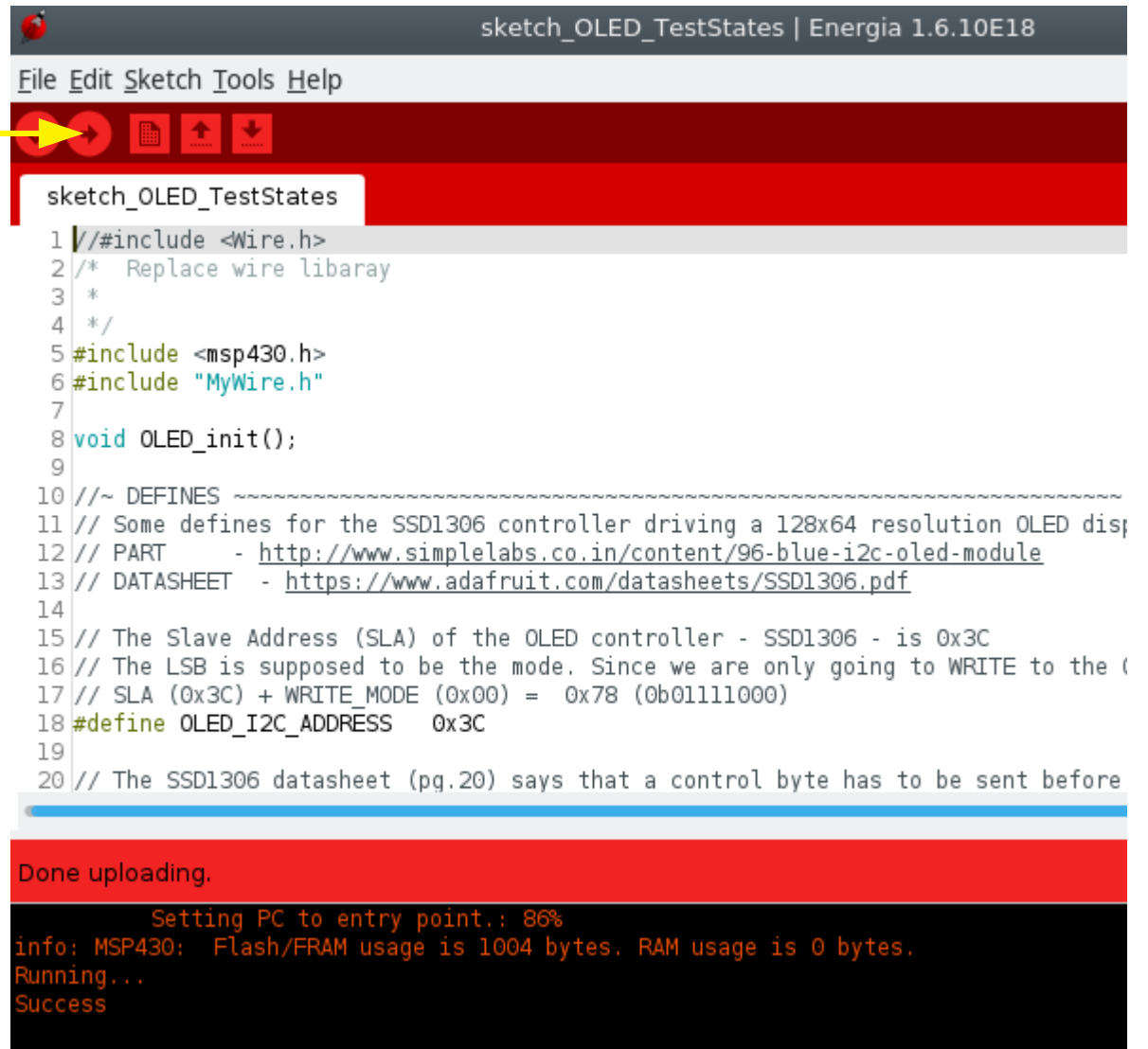


Then save as it's own sketch



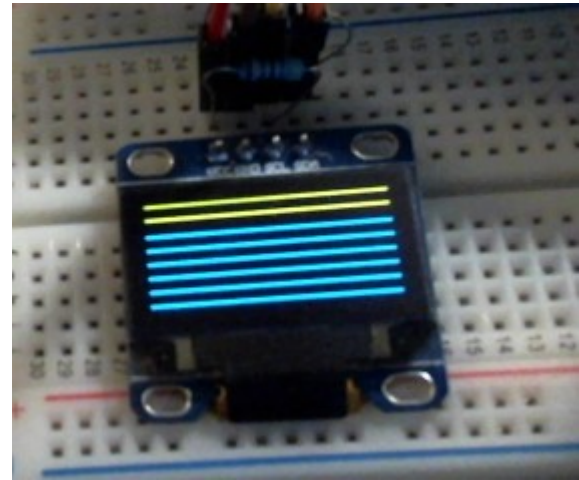
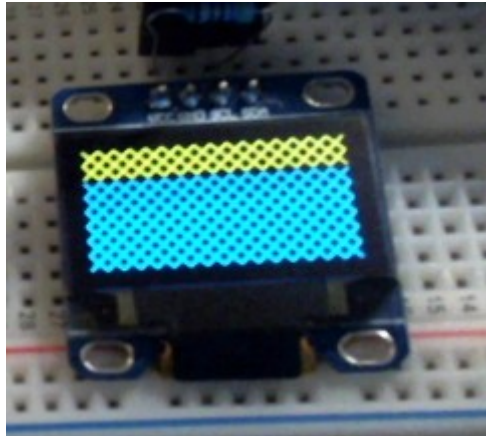
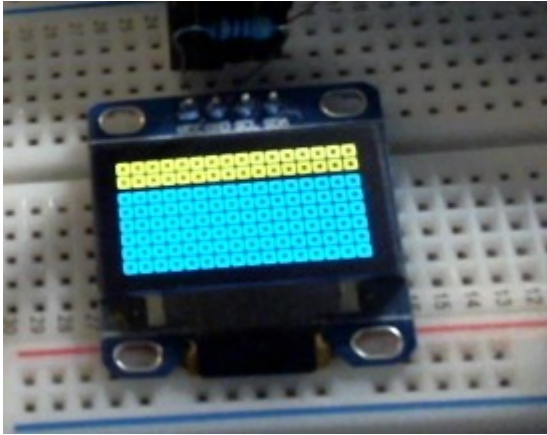
Compile and upload  
the Example sketch

then Run on the  
OLED Display



```
sketch_OLED_TestStates | Energia 1.6.10E18
File Edit Sketch Tools Help
sketch_OLED_TestStates
1 // #include <Wire.h>
2 /* Replace wire library
3 *
4 */
5 #include <msp430.h>
6 #include "MyWire.h"
7
8 void OLED_init();
9
10 // ~ DEFINES ~~~~~
11 // Some defines for the SSD1306 controller driving a 128x64 resolution OLED display
12 // PART - http://www.simplelabs.co.in/content/96-blue-i2c-oled-module
13 // DATASHEET - https://www.adafruit.com/datasheets/SSD1306.pdf
14
15 // The Slave Address (SLA) of the OLED controller - SSD1306 - is 0x3C
16 // The LSB is supposed to be the mode. Since we are only going to WRITE to the (
17 // SLA (0x3C) + WRITE_MODE (0x00) = 0x78 (0b01111000)
18 #define OLED_I2C_ADDRESS 0x3C
19
20 // The SSD1306 datasheet (pg.20) says that a control byte has to be sent before
Done uploading.
Setting PC to entry point.: 86%
info: MSP430: Flash/FRAM usage is 1004 bytes. RAM usage is 0 bytes.
Running...
Success
```

Each second a different display is shown



Etc...

## Initialize the OLED

```
93  
94   while (1)  
95   {  
96       // I2C  
97       WTx.beginTransmission(OLED_I2C_ADDRESS);  
98       WTx.write(OLED_CONTROL_BYTE_CMD_STREAM);  
99       WTx.write(OLED_CMD_SET_COLUMN_RANGE);  
100      WTx.write(0x00);  
101      WTx.write(0x7F);  
102      WTx.write(OLED_CMD_SET_PAGE_RANGE);  
103      WTx.write(0);  
104      WTx.write(0x07);  
105      WTx.endTransmission();  
106  
107
```



```

108 for(uint16_t i=0;i<1024;i++){
109     WTx.beginTransmission(OLED_I2C_ADDRESS);
110     WTx.write(OLED_CONTROL_BYTE_DATA_STREAM);
111
112     for (uint8_t x=0; x<16; x++) {
113         switch(MyState){
114             case 0:
115                 // WTx.write(0b11000001);
116                 MyPattern = 0b11000001;
117                 break;
118             case 1:
119                 // WTx.write(0x81);
120                 MyPattern = 0x81;
121                 break;
122             case 2:
123                 // WTx.write(0x02);
124                 MyPattern = 0x02;
125                 break;
126             case 3:
127                 // WTx.write(pattern1[x]);
128                 MyPattern = pattern1[x];
129                 break;
130             case 4:
131                 // WTx.write(pattern2[x]);
132                 MyPattern = pattern2[x];
133                 break;
134             default:
135                 break;
136         }
137         WTx.write(MyPattern);
138     i++;
139     }
140     i--;
141     WTx.endTransmission();
142

```

This is the code that writes each new pattern to the OLED

This is done once per second (using delay function .. next slide)

Write a single byte to the OLED

```
140     i--;
141     WTx.endTransmission();
142
143 }
144     if (MyState++>4) MyState=0;
145
146     // for(unsigned long int del=0; del<100000; del++);
147     //delay(1);
148     // for (MyWait=0; MyWait<63000; MyWait++);
149     WTx.MyDelay();
150
151     P1OUT ^= BIT0; // P1.0 = toggle (xor)
152 }
153
```

Increment the State

Delay function



# Program Logic

Use hardware to power down while waiting for event end interrupt

Send char to display – power down to wait

Delay – power down for period of delay

Program possesses the Foreground priority – Always in control (100% CPU)

Background is Low Power Mode only while waiting for completion event

Bad use of ISR only LPM while waiting

Poor programming, somewhere between Arduino 100% CPU utilization using LPM programming only when waiting

Use Timer\_A to generate wait in Low Power Mode  
Low power mode is only enabled while waiting.....

## Code Analysis: MyWire library

```
88 void MyWire::MyDelay(void)
89 {
90     TA0CTL0 |= CCIE;           // TACCR0 interrupt enabled
91     TA0EX0 |= TAIDEX_3;       // SMCLK/8/4 = 31250 Hz
92     TA0CCR0 = 31250;          // 1 per second
93     TA0CTL = TASSEL_2 | MC_1 | ID_3; // SMCLK/8 = 125K , UP mode
94
95     // go to Standby
96     __bis_SR_register(LPM0_bits | GIE); // go to sleep
97 }
98
99 // Timer A0 interrupt service routine
100 #pragma vector = TIMER0_A0_VECTOR
101 __interrupt void MyWire::Timer_A_ISR (void)
102 {
103     // P1OUT ^= BIT0;
104     __bic_SR_register_on_exit(LPM0_bits); // Exit LPM0 Wake up
105 }
106
107
```

`__bic_SR_register_on_exit(LPM0_bits);` Exits the ISR with LPM0 and Interrupt turned off  
'bit clear'

