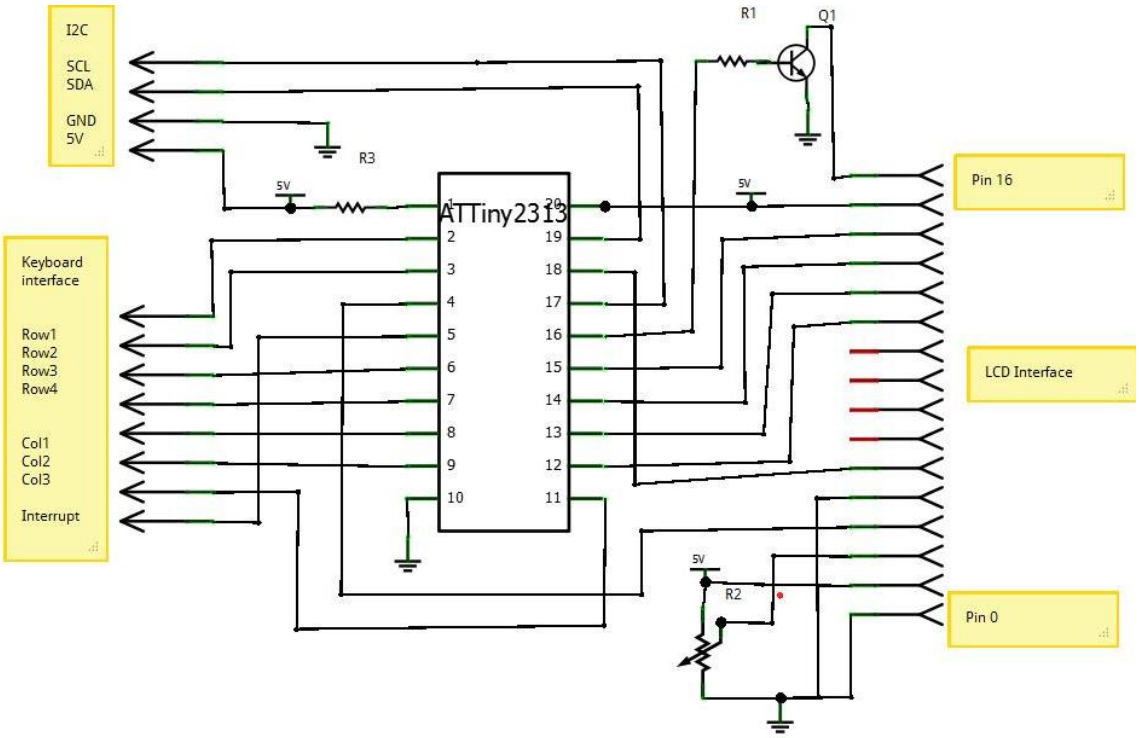
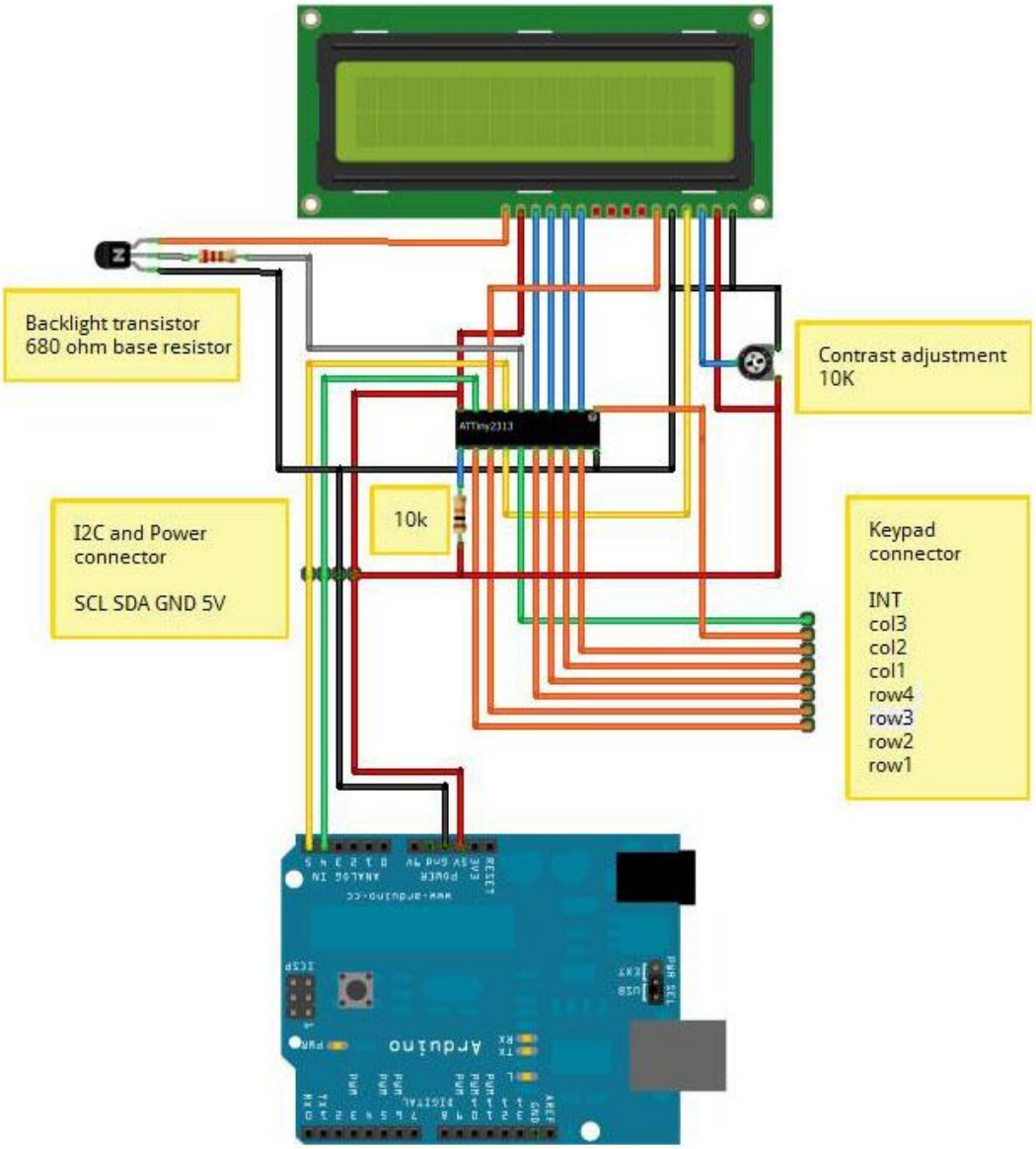
ATTiny2313 LCD and Keypad controller

This document describes the I2C LCD controller – designed to control LCD displays with the HD4470 parallel interface.

An ATIny2313 micro controller was chosen for this project. The code for this stretches the ATTiny2313 to its limit in terms of size

1. Wiring (sorry amateur diagram – still learning Fritzing).





2. I2C commands
   The default address of the I2C chip is 0x12 – this is a 7 bit address and is suitable for the Wire library in the Arduino. A full eight bit address would be 0x24 (read) and 0x25(write).

   a. Send characters to the LCD
      Just transmit characters to the device. The receive buffer is 32 characters long, if this is exceeded in one message then the extra will be discarded.
      It takes around 60 micro seconds to process each character, so a message with 32 characters will need a delay of 2 milli seconds before any further messages are sent.

      Character codes 0xfe and 0xff (decimal 254 and 255) are special codes, so to send these we need to precede them with 0xfe.

      Example Arduino code:

```
#include <Wire.h>

void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
  sendStr("Hello World");
}

void loop()
{
}

void sendStr(char* b)
{
  Wire.beginTransmission(0x12); // transmit to device 12
  while (*b)
  {
    if (*b == 0xfe || *b == 0xff) Wire.send(0xfe);
    Wire.send(*b++);              // sends one byte
  }
  Wire.endTransmission();   // stop transmitting
  delay(2);
}
```

b. Commands (HD4470)

All the HD4470 commands are prefixed by 0xfe (decimal 254).
For full details please see the HD4470 data sheet.
For command 0x01 and 0x02 add a delay of 2mS after the command. For all other commands a delay of 50uS is sufficient.
Some of the supported commands are:

0xfe, 0x01	Clears entire display and sets cursor to the beginning
0xfe, 0x02	Clears entire display and undoes any shift
0xfe, 0x08	sets display options – OR with 0x04 display on /off 0x02 Cursor on / off 0x01 Cursor blinking on / off
0xfe, 0x10	Shift display or cursor – OR with 0x08 Cursor = 0, display = 1 0x04 shift right = 1, shift left = 0
0xfe, 0x40	Set into programmable character mode – OR with CGRAM address. Subsequent data sent programmes the characters and does not display on the LCD.
0xfe, 0x80	Set back to 'normal' character mode and position the cursor OR with the cursor address. See the HD4470 datasheet for Information on how to work out cursor addresses. Subsequent data sent is displayed on the LCD

Example Arduino code:

```
#include <Wire.h>
```

```
void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
  clearLCD();
  sendStr("Hello World");
}

void loop()
{
}

void sendStr(char* b)
{
  Wire.beginTransmission(0x12); // transmit to device 12
  while (*b)
  {
    if (*b == 0xfe || *b == 0xff) Wire.send(0xfe);
    Wire.send(*b++);              // sends one byte
  }
  Wire.endTransmission();   // stop transmitting
  delay(2);
}

void clearLCD()
{
  Wire.beginTransmission(0x12); // transmit to device 12
  Wire.send(0xfe);          // signal command follows
  Wire.send(0x01);          // send the command
  Wire.endTransmission();   // stop transmitting
  delay(2);
}
```

c. Extended commands

All the extended commands are prefixed with 0xff (decimal 255).

i. Backlight

0xff, 0x01, BL BL =0 backlight off, otherwise on

Arduino code:

```
void backlight(byte on)
{
    Wire.beginTransmission(0x12); // transmit to device 12
    Wire.send(0xFF);              // sends command flag
    Wire.send(0x01);              // backlight
    Wire.send(on);              // sends on/off flag
    Wire.endTransmission();   // stop transmitting
    delay(1);          // always a good idea to give the
                // I2C device a chance to catch up.
}
```

ii. EEPROM

0xff, 0x02, addr	Reads EEPROM address addr
0xff, 0x03, addr, value	Write value into EEPROM address addr
0xff, 0x04, addr	Display zero terminated string at EEPROM address addr.

Arduino code:

```
byte readEEPROM(byte addr)
{
  Wire.beginTransmission(0x12); // transmit to device 12
  Wire.send(0xff);            // signal command follows
  Wire.send(0x02);            // send command read EEPROM
  Wire.send(addr);            // send the address
  Wire.endTransmission();   // stop transmitting
  delay(1);                // give it a chance
  Wire.requestFrom(0x12,1);   // ask for the byte
  return Wire.receive();    // and return it
}
```

iii. Reset

0xff, 0xf0 Reset EEPROM to defaults and reset the chip

0xff, 0xf1 reset the chip – reading current values from EEPROM

Arduino code:

```
void changeI2CAddress(byte value)
{
  Wire.beginTransmission(0x12); // transmit to device 12
  Wire.send(0xff);            // signal command follows
  Wire.send(0x03);            // send the command write
                              // EEPROM
  Wire.send(0x00);            // EEPROM addr is I2C address
  Wire.send(value);           // send the new I2C address
  Wire.endTransmission();   // stop transmitting
  delay(1);
  Wire.beginTransmission(0x12); // transmit to device 12
  Wire.send(0xff);            // signal command follows
  Wire.send(0xf1);            // send the reset
  Wire.endTransmission();   // stop transmitting
  delay(4);
  // the address is now changed and you need to use the new
address
}
```

iv. Keypad interface

The keypad interface has a buffer of 15 keys. If there are any key presses in the buffer the interrupt line is high, it goes low when the buffer is emptied. The default mapping of the keys can be changed – see Keyboard mapping section.

All key presses are debounced for 30mS – debounce period can be modified at EEPROM address 6.

Commands:

0xff, 0x10 Returns one byte – number of keys in buffer

0xff, 0x11 Returns next key in buffer, and removes it from the buffer

0xff, 0x12 returns the keycode for a key currently held down

0xff, 0x13	clears the buffer
0xff, 0x14, n	Read n bytes from the keypad buffer
0xff, 0x15	Interrupt pin off
0xff, 0x16	Interrupt pin on

Arduino code:

```
byte readKeyp()
{
    Wire.beginTransmission(0x12); // transmit to device 12
    Wire.send(0xFF);              // sends command flag
    Wire.send(0x11);              // read next key
    Wire.endTransmission();   // stop transmitting
    delay(2);
    Wire.requestFrom(0x12,1);
    return Wire.receive();  // get the next char in the
                        // buffer
                        // note zero is returned if none
}
```

3. Modifying startup parameters
   EEPROM addresses 2-5 contain the bytes which are sent to the LCD at power up.
   Modifying these values allow you to customise the LCD (e.g. for a one line LCD).
   The default is set up for a two or four line LCD.
4. Hardware reset settings
   Connecting the Column 1 pin (PortD4 ATTiny pin 8) to ground during power up will reset all of the EEPROM setting to the default. This is useful for example if you re-program the I2C address and then forget it.
5. EEProm layout

Address	Default value	Meaning
0	0x12	I2C address
1	0x07	Code version number
2	0x28	LCD initialisation – LCD function mode
3	0x0e	LCD initialisation – Cursor direction
4	0x06	LCD initialisation – entry mode
5	0x01	LCD initialisation - clear
6	0x1E	Keyboard debounce time in mS
7	0	Future use
8 - 20	Keyboard map	Keyboard map
21 - 93	“JGC I2C V0.7”	Initial string (currently only 13 characters used).

6. Keypad mapping
   EEPROM address 8 is the value used for no key in the buffer, default is 0.

No key in buffer – returns default 0.

EEPROM addr	Row	Column	Keycode	Returns default value
9	1	1	1	0x31 ‘1’
10	1	2	2	0x32 ‘2’
11	1	3	3	0x33 ‘3’

12	2	1	4	0x34 ‘4’
13	2	2	5	0x35 ‘5’
14	2	3	6	0x36 ‘6’
15	3	1	7	0x37 ‘7’
16	3	2	8	0x38 ‘8’
17	3	3	9	0x39 ‘9’
18	4	1	10	0x2A ‘*’
19	4	2	11	0x30 ‘0’
20	4	3	12	0x23 ‘#’