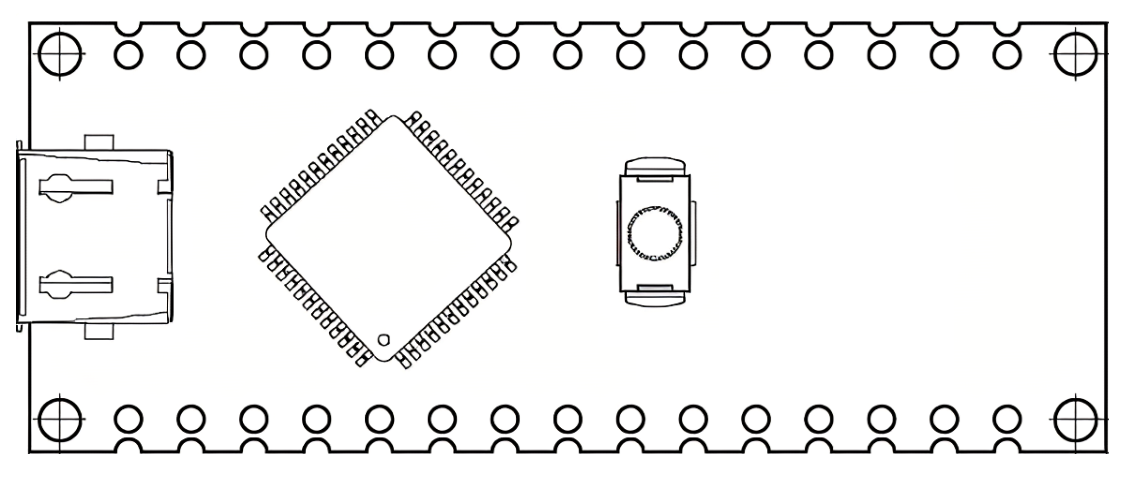
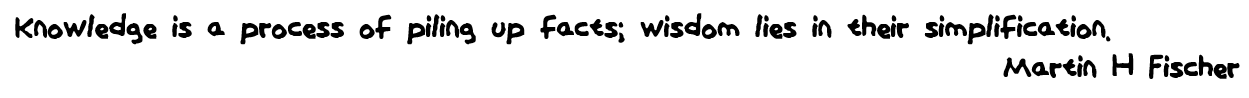
**Experimenting with Arduino comms,**

**LCD, Keypad and LEDs.**



**Shows how to**

* **Switch on and off LEDs**
* **Get data from Arduino to C# Windows app**
* **Talk from C# Windows App to Arduino**



Contents

[1. Introduction 3](#_Toc149743768)

[2. Parts list 3](#_Toc149743769)

[3. Wiring Diagram 3](#_Toc149743770)

[4. Libraries 3](#_Toc149743771)

[5. Find Modules Addresses 4](#_Toc149743772)

[6. C# Windows App 4](#_Toc149743773)

[7. Arduino C++ Code 6](#_Toc149743774)

[7.1. Switching LEDs 6](#_Toc149743775)

[7.2. LCD Display 7](#_Toc149743776)

[7.2.1. Library 7](#_Toc149743777)

[7.2.2. I²C Address 7](#_Toc149743778)

[7.2.3. Define the LCD 7](#_Toc149743779)

[7.2.4. Setup() 7](#_Toc149743780)

[7.2.5. Loop() 8](#_Toc149743781)

[7.3. Keyboard 9](#_Toc149743782)

[7.3.1. Library 9](#_Toc149743783)

[7.3.2. Keypad pinouts 9](#_Toc149743784)

[7.3.3. Define the keyboard 9](#_Toc149743785)

[7.3.4. Setup() 9](#_Toc149743786)

[7.3.5. Loop() 10](#_Toc149743787)

[8. Appendix A – Find I2C addresses 11](#_Toc149743788)

[10. Appendix B – Arduino Code 12](#_Toc149743789)

# Introduction

In this project we add some LEDS, a keypad and an LCD display to an Arduino Nano. We will send data from the Nano to a C# App running on a PC/Laptop. We will also send data from the app to the Nano and display it on the LCD.

The results will give you an idea of how to connect things and give you the ideas on how to modify this for your own use.

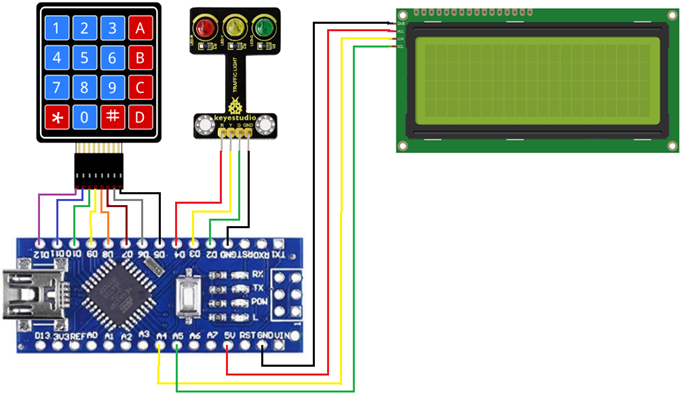
# Parts list

For this project we will use:

* Arduino Nano
* Keyestudio Traffic Light Module
* I2C 4x20 LCD screen
* 4×4 Matrix Keypad 16 Key Thin & Flexible with Cable

You may have different items but may find the pinouts the same. Below you will see what these items look like.

# Wiring Diagram



# Libraries

To make this work a few libraries are needed. These are discussed in the sections below.

Once installed these are found @ C:\Users\Your\_Name\Documents\Arduino\libraries

# Find Modules Addresses

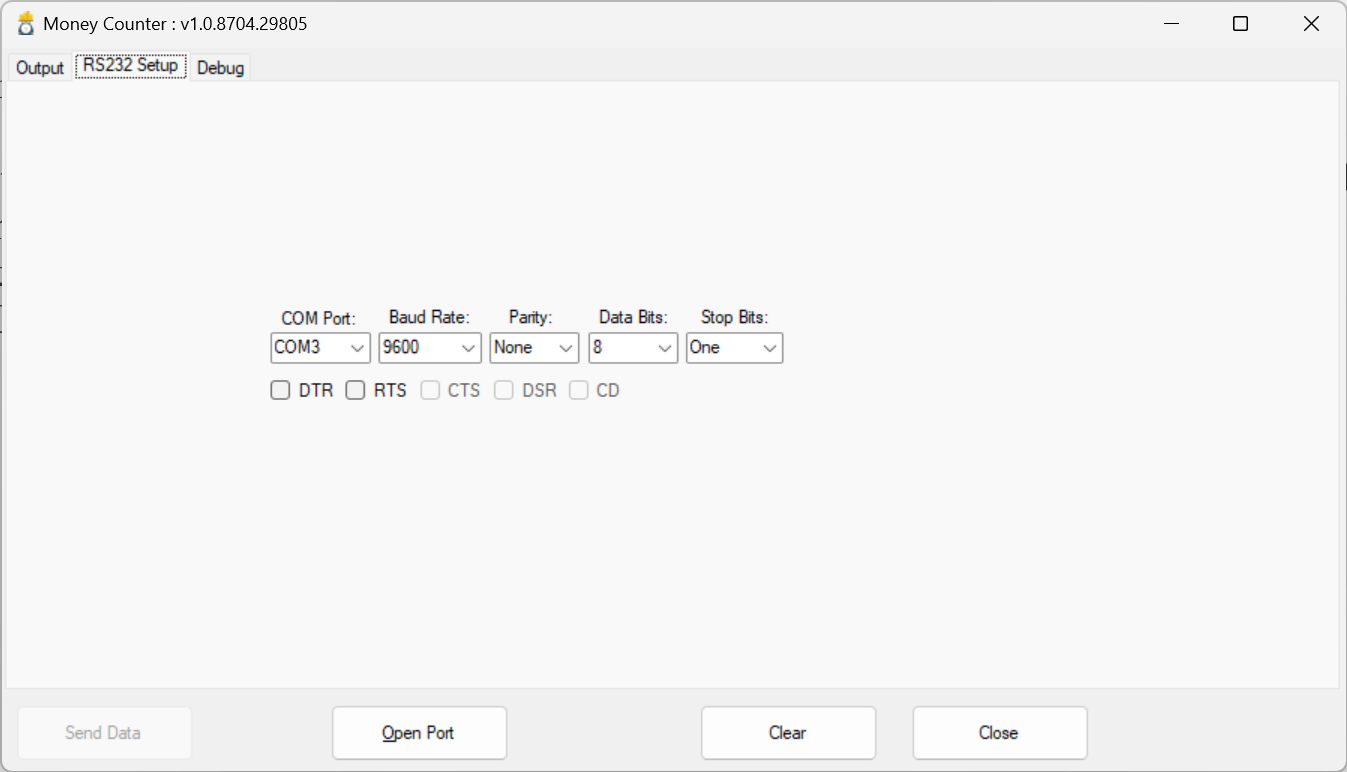
If you want you can find the address of each correctly wired I2C device on your circuit.

You will find a code sample in Appendix A that you can use to scan for all the I2C devices on your circuit it will output the address to the Arduino IDE serial monitor.

# C# Windows App

Here we use another app that we are working on and just add enough into it to make it work for this scenario. You may want to start with this app and change it to meet your means.

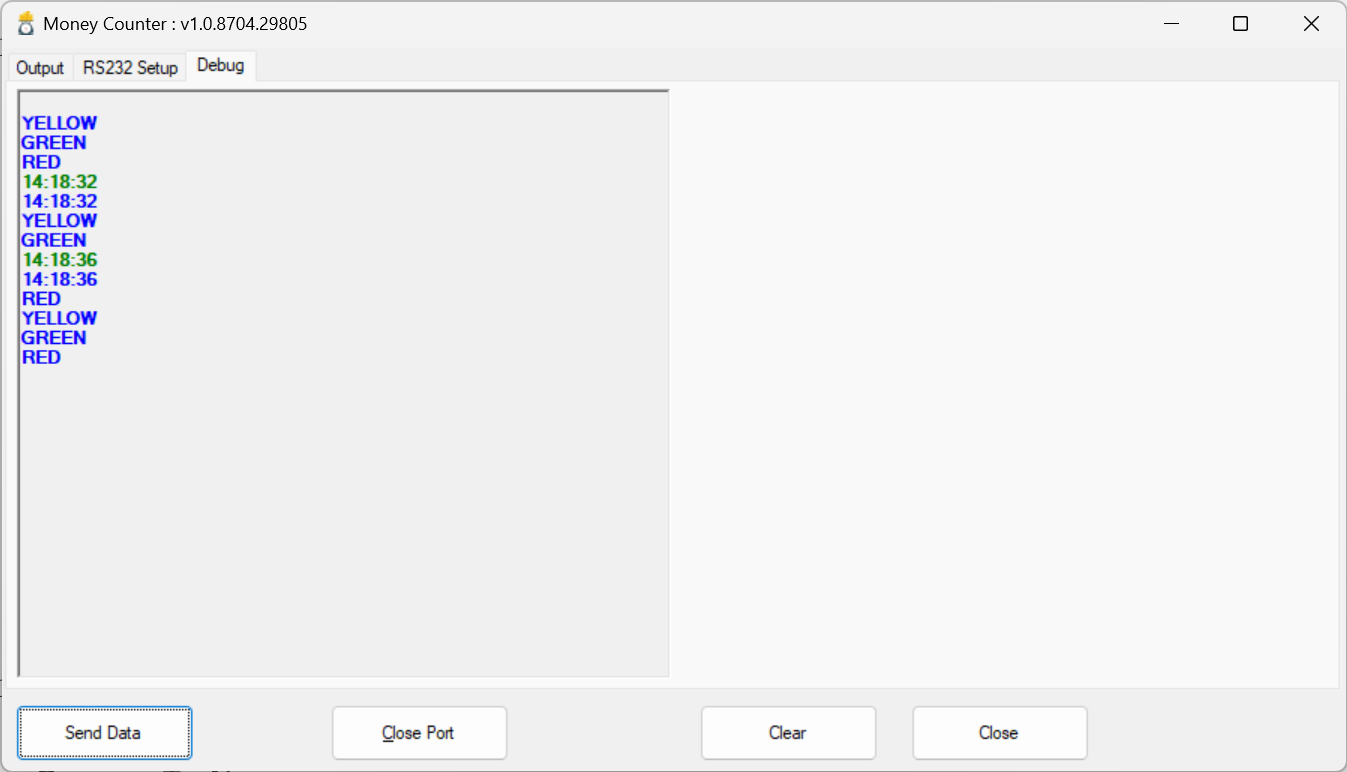
Under the “RS232 Setup” tab you will find the settings to use to connect to your Nano.



Once you have set the menus to what you need the click on the Open Port button to connect to the Nano.

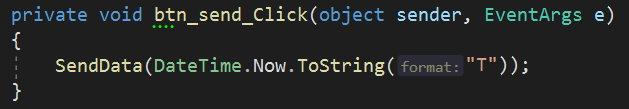
Under the debug tab you will find all the comms between the Nano and the App.

* Blue = From Nano
* Green = To Nano

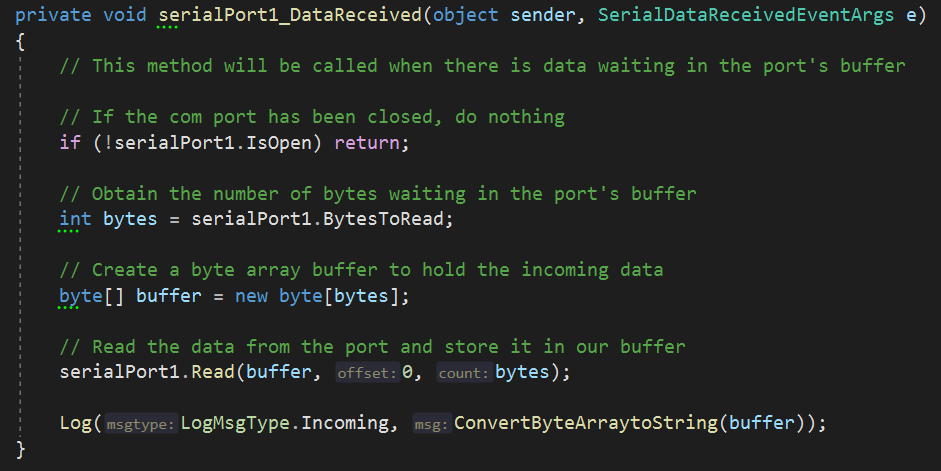


To send data to the Nano click on the “Send Data” button. This will send over the time which will be shown on the 3rd line down.

In the code in file Form1.cs you will find the following event. The send data function sends the data to the Nano. You can change the parameter to what you need to send.



Data coming in from the Arduino will be dealt with in the file “SerialPort.cs” in the function as shown below. You may want to change the code here to do something with the data coming in.



By tickling the DTR line you will be able to reboot the Arduino.

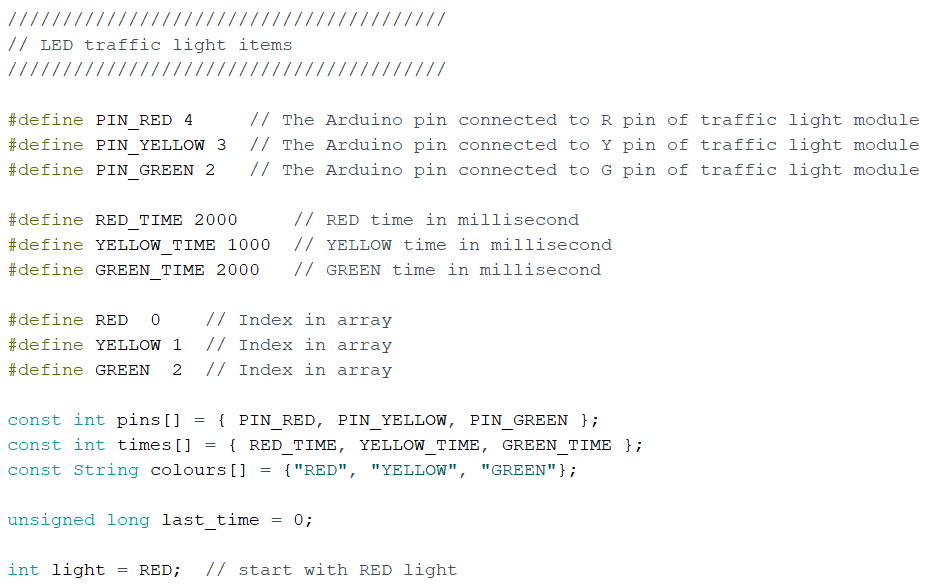
# Arduino C++ Code

The relevant parts are described next as individual sections. The code is in Appendix B

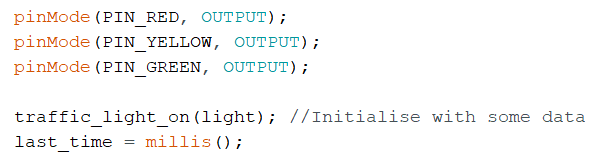
## Switching LEDs

Here we will use a small module that has three LEDs in it.

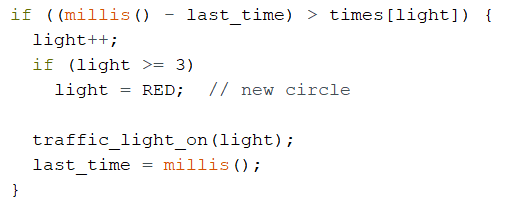
Declarations



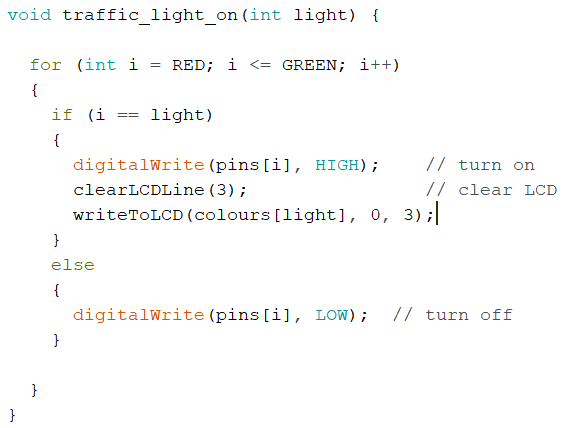
Under Setup() we add



Under Loop() we add the following to do the time an LED should stay on and the call the function traffic\_light\_on()



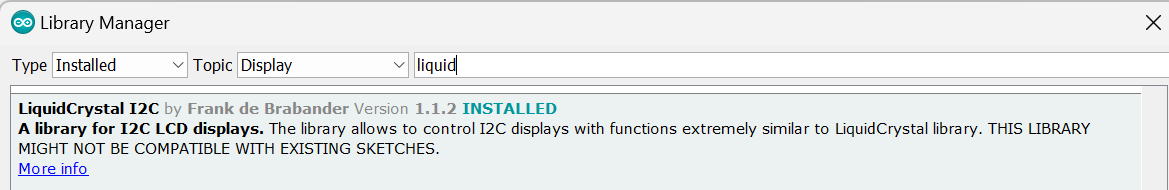
Under traffic\_light\_on() switch the LEDs on (high) and off (low)



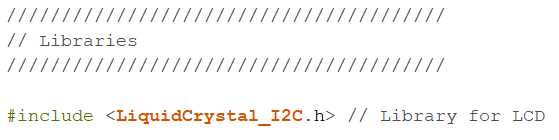
## LCD Display

### Library

We will be using the library as shown below:

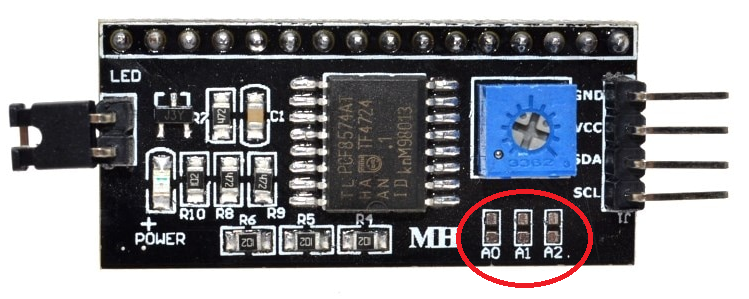


Remember to add this to the code as follows:



### I²C Address

Some I²C LCD module have an address selector solder pad.



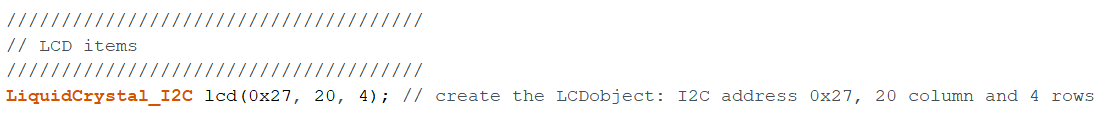
They are usually labelled with A0, A1, and A2. The following table show you how to interpret the selector. “1” = Not Connected, “0” = Connected.

|  |  |  |  |
| --- | --- | --- | --- |
| **A0** | **A1** | **A2** | **HEX Address** |
| 1 | 1 | 1 | 27 |
| 0 | 1 | 1 | 26 |
| 1 | 0 | 1 | 25 |
| 0 | 0 | 1 | 24 |
| 1 | 1 | 0 | 23 |
| 0 | 1 | 0 | 22 |
| 1 | 0 | 0 | 21 |
| 0 | 0 | 0 | 20 |

We used the HEX Address 0x27 in the example definition below.

### Define the LCD

In the code you will need to add the following:



### Setup()

Initialise the LCD as follows in the setup section.

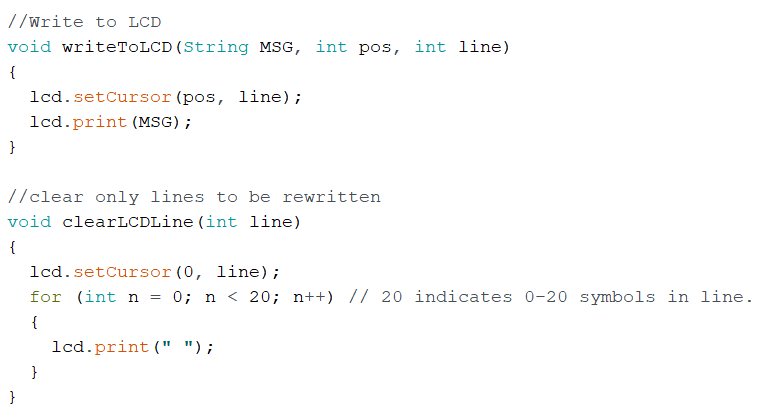


### Loop()

Anywhere in the loop that you want text to be written to the screen you can do the following. There are many variations of this that you may choose.



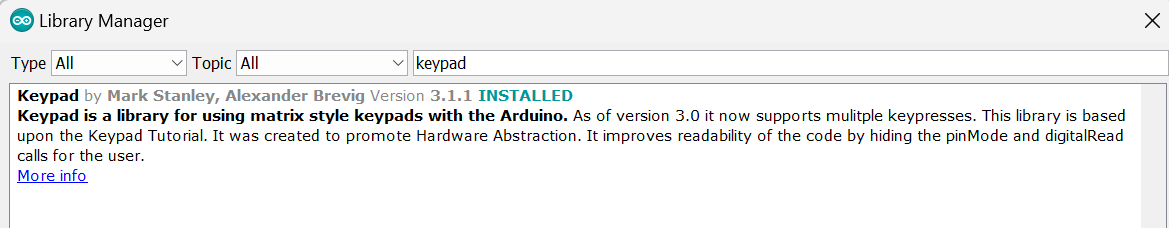
We have created two function to handle the clearing and writing to the screen. This keeps our code neater. You may choose to do it differently.



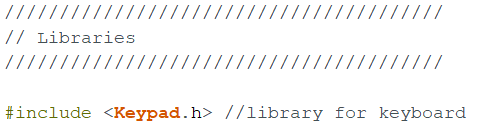
## Keyboard

### Library

We will be using the library as shown below:

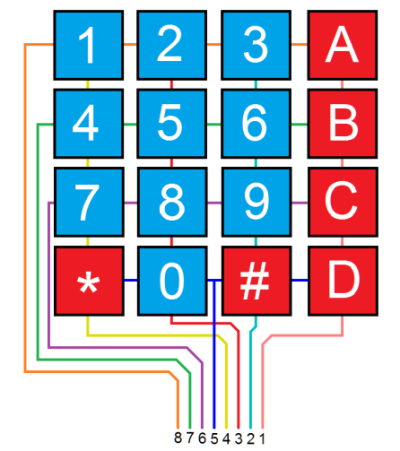


Remember to add this to the code as follows:



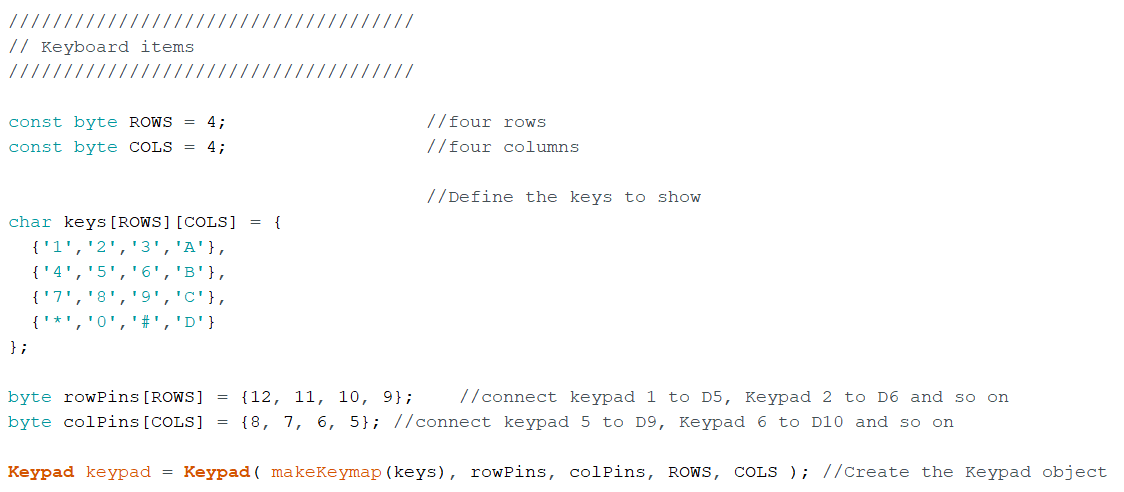
### Keypad pinouts

We are using the keypad as shown below with its respective pinouts:



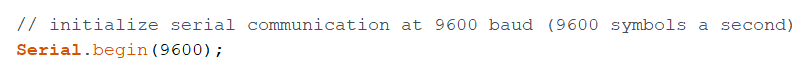
### Define the keyboard

In the code you will need to add the following:



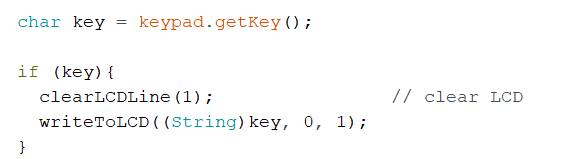
### Setup()

Initialise the RS232 as we use this to grab the data



### Loop()

Add the following, note that in the if (key) you need to add the code you need, here you see what our code uses:



# Appendix A – Find I2C addresses

#include <Wire.h>

void setup() {

**Serial**.begin (115200);

  // wait for serial port to connect

  while (!**Serial**)

    {

    }

**Serial**.println ();

**Serial**.println ("I2C scanner. Scanning ...");

  byte count = 0;

  Wire.begin();

  for (byte i = 8; i < 120; i++)

  {

    Wire.beginTransmission (i);

    if (Wire.endTransmission () == 0)

      {

**Serial**.print ("Found address: ");

**Serial**.print (i, DEC);

**Serial**.print (" (0x");

**Serial**.print (i, HEX);

**Serial**.println (")");

      count++;

      delay (1); // maybe unneeded?

      } // end of good response

  } // end of for loop

**Serial**.println ("Done.");

**Serial**.print ("Found ");

**Serial**.print (count, DEC);

**Serial**.println (" device(s).");

}  // end of setup

void loop() {}

# Appendix B – Arduino Code

////////////////////////////////////////

// Libraries

////////////////////////////////////////

#include <LiquidCrystal\_I2C.h> // Library for LCD

#include <Keypad.h> //library for keyboard

////////////////////////////////////////

// LED traffic light items

////////////////////////////////////////

#define PIN\_RED 4 // The Arduino pin connected to R pin of traffic light module

#define PIN\_YELLOW 3 // The Arduino pin connected to Y pin of traffic light module

#define PIN\_GREEN 2 // The Arduino pin connected to G pin of traffic light module

#define RED\_TIME 2000 // RED time in millisecond

#define YELLOW\_TIME 1000 // YELLOW time in millisecond

#define GREEN\_TIME 2000 // GREEN time in millisecond

#define RED 0 // Index in array

#define YELLOW 1 // Index in array

#define GREEN 2 // Index in array

const int pins[] = { PIN\_RED, PIN\_YELLOW, PIN\_GREEN };

const int times[] = { RED\_TIME, YELLOW\_TIME, GREEN\_TIME };

const String colours[] = {"RED", "YELLOW", "GREEN"};

unsigned long last\_time = 0;

int light = RED; // start with RED light

//////////////////////////////////////

// LCD items

//////////////////////////////////////

LiquidCrystal\_I2C lcd(0x27, 20, 4); // create the LCDobject: I2C address 0x27, 20 column and 4 rows

/////////////////////////////////////

// Keyboard items

/////////////////////////////////////

const byte ROWS = 4; //four rows

const byte COLS = 4; //four columns

//Define the keys to show

char keys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},

{'\*','0','#','D'}

};

byte rowPins[ROWS] = {12, 11, 10, 9}; //connect keypad 1 to D5, Keypad 2 to D6 and so on

byte colPins[COLS] = {8, 7, 6, 5}; //connect keypad 5 to D9, Keypad 6 to D10 and so on

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS ); //Create the Keypad object

void setup() {

lcd.init(); //initialize the lcd

lcd.backlight(); //open the backlight

pinMode(PIN\_RED, OUTPUT);

pinMode(PIN\_YELLOW, OUTPUT);

pinMode(PIN\_GREEN, OUTPUT);

traffic\_light\_on(light); //Initialise with some data

last\_time = millis();

// initialize serial communication at 9600 baud (9600 symbols a second)

Serial.begin(9600);

lcd.setCursor(0, 0); // move cursor the first row

lcd.print("LCD 20x4"); // print message at the first row

lcd.setCursor(0, 1); // move cursor to the second row

lcd.print("I2C Address: 0x27"); // print message at the second row

lcd.setCursor(0, 2); // move cursor to the third row

lcd.print("Light on"); // print message at the third row

}

// the loop function runs over and over again forever

void loop() {

if ((millis() - last\_time) > times[light]) {

light++;

if (light >= 3)

light = RED; // new circle

traffic\_light\_on(light);

last\_time = millis();

}

char key = keypad.getKey();

if (key){

clearLCDLine(1); // clear LCD

writeToLCD((String)key, 0, 1);

}

}

void traffic\_light\_on(int light) {

// Serial.println(light);

for (int i = RED; i <= GREEN; i++)

{

if (i == light)

{

digitalWrite(pins[i], HIGH); // turn on

clearLCDLine(3); // clear LCD

writeToLCD(colours[light], 0, 3);

//Serial.println(colours[light]);

}

else

{

digitalWrite(pins[i], LOW); // turn off

}

}

}

//Write to LCD

void writeToLCD(String MSG, int pos, int line)

{

lcd.setCursor(pos, line);

lcd.print(MSG);

Serial.println(MSG);

}

//clear only lines to be rewritten

void clearLCDLine(int line)

{

lcd.setCursor(0, line);

for (int n = 0; n < 20; n++) // 20 indicates 0-20 symbols in line.

{

lcd.print(" ");

}

}

/\*

SerialEvent occurs whenever new data comes in the hardware serial RX. This

routine is run between each time loop() runs, so using delay inside loop can

delay response. Multiple bytes of data may be available. We only send data

once.

\*/

void serialEvent()

{

char inChar;

String DataString = ""; // a String to hold incoming data

DataString.reserve(21); // reserve 21 bytes for the inputString:

delay(100); //Add delay not to miss first byte

while (Serial.available())

{

// get the new byte:

inChar = (char)Serial.read();

// add it to the DataString:

DataString += inChar;

}

clearLCDLine(2); // clear LCD

writeToLCD(DataString, 0, 2); // write data

DataString = ""; // Empty string

}