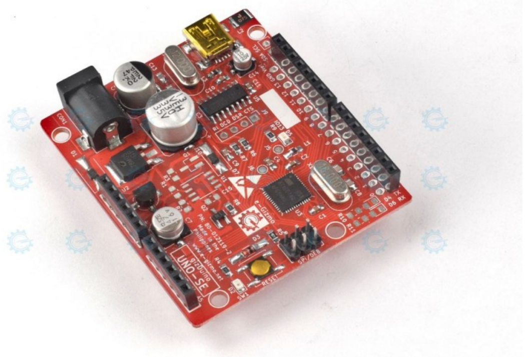
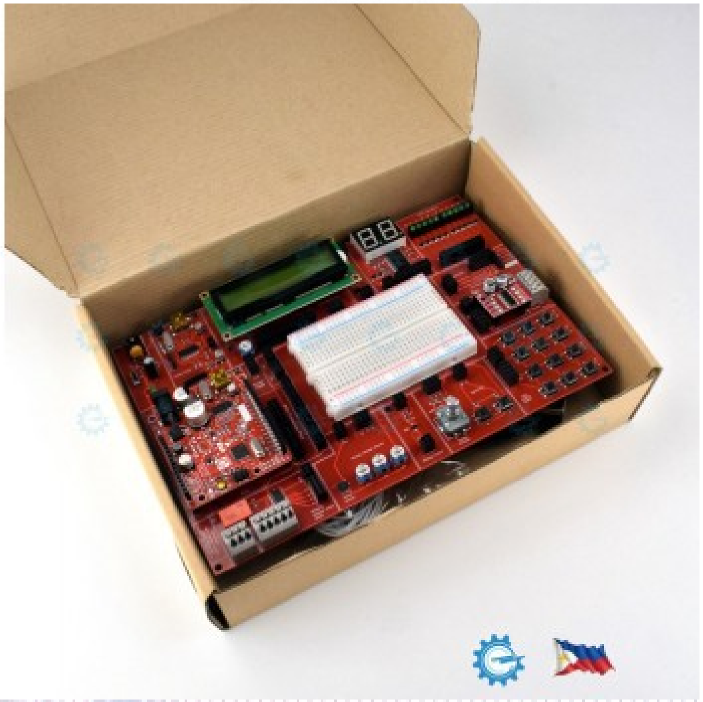
Universal Micro-controller Trainer Board

With gizDuino SE included.

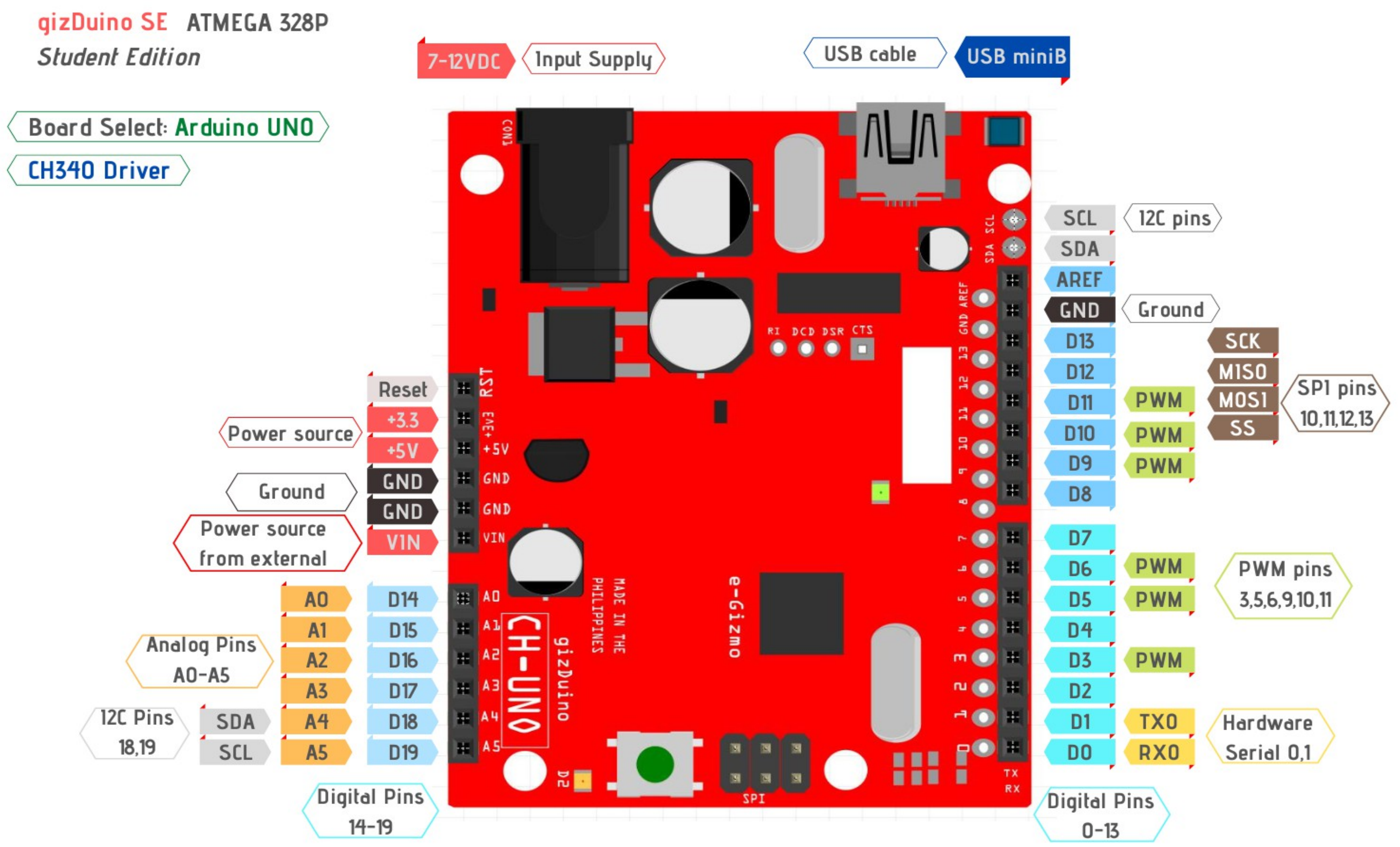




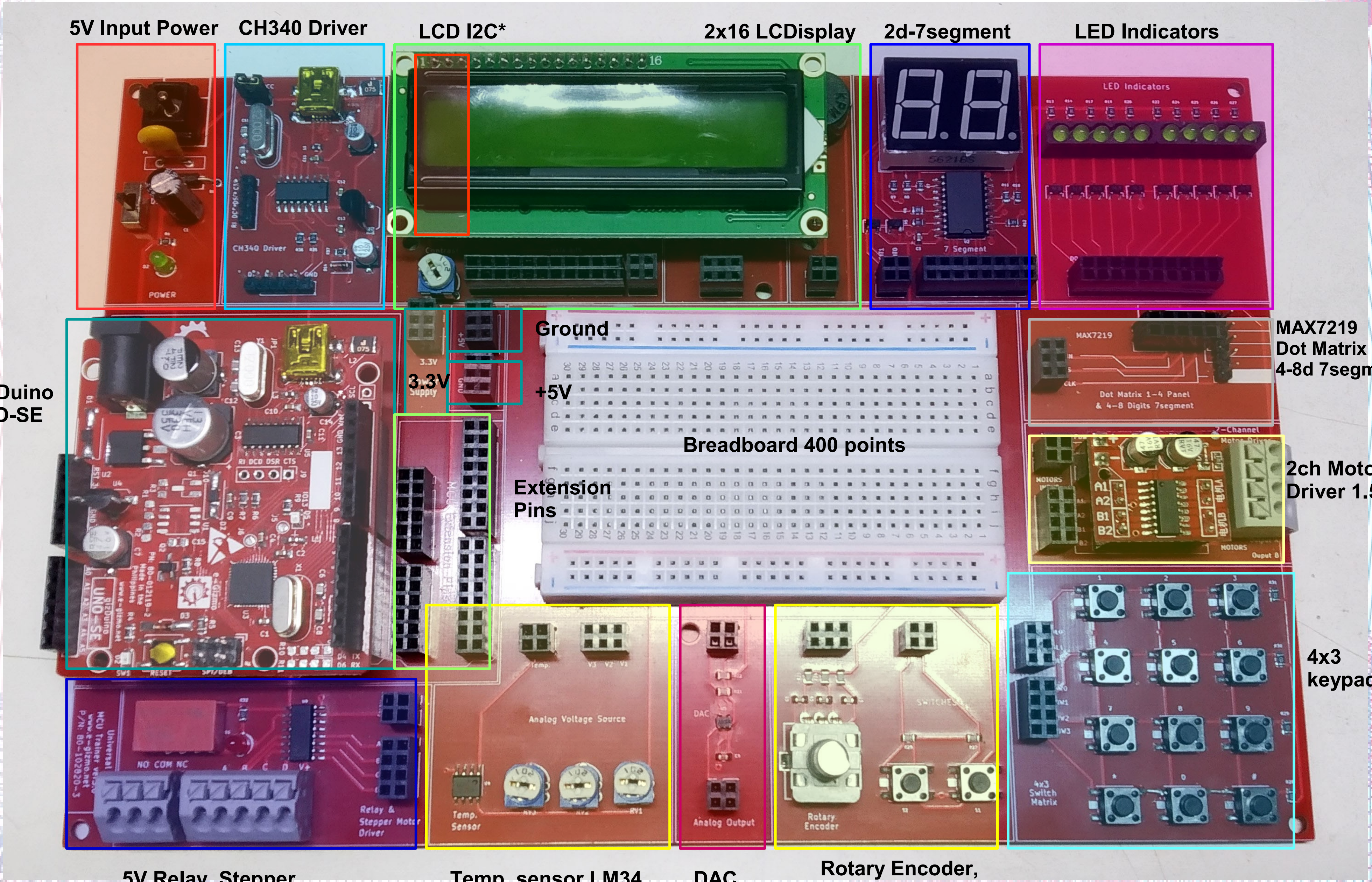
Suitable for Online-classes

GizDuino SE (Student Edition) as a MAIN controller

Parts and descriptions



PARTS


5V Input Power
CH340 Driver
LCD I2C*
2x16 LCDisplay
2d-7segment
LED Indicators
GizDuino UNO-SE
Ground
3.3V
+5V
MAX7219 Dot Matrix & 4-8d 7segment
Breadboard 400 points
Extension Pins
2ch Motor Driver 1.5A
4x3 keypad
5V Relay, Stepper Driver Unipolar
Temp. sensor LM34, Analof Voltage Source
DAC
Rotary Encoder, Switches

Package included

- gizDuino UNO-SE with Cable
- 5V Adaptor
- 20-jumper M-M wires 20cm
- Stepper Motor
- 6pc brass stud and screws for stand

Specifications

Input supply: 5VDC

Modules on board: 16

Microcontroller compatible

- gizDuino UNO-SE (Arduino UNO)
- gizDuino LIN-UNO
- gizDuino V (328/168)
- gizDuino Plus (164,324,644)
- gizDuino X ATmega1281

- gizDuino MINI (88,168,328)
- gizDuino miniUSB(168/328)

On board Modules:

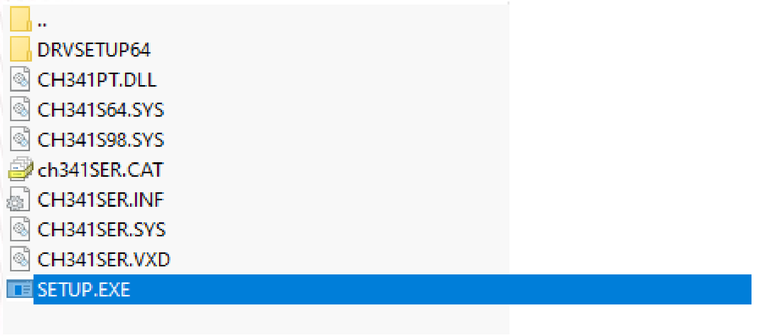
- H340 Driver (For gizDuino mini168p/328p. +mini164p/324p/644p, Arduino mini/pro)
- 2x16 LCD Display Green
- For 2x16 LCD with I2C module socket connector*
- DS1307 Real-Time Clock module
- Passive Buzzer
- 2-digit 7 segment display
- 10 LED indicators*
- For MAX7219 Dot Matrix 1-4 Panel and 4-8 Digits 7segment socket connector*(module sold separately)
- 2-Channel DC motor driver 1.5A*(dc motor not included)
- 4x3 Switch Matrix
- 2-Push buttons
- Rotary Encoder
- DAC (Digital-to-Analog converter)
- Analog Voltage Source
- Temperature Sensor LM34
- Stepper Motor Driver ULN2003A with Unipolar stepper motor
- 5V relay

Legend: *New Features

CH340 Manual Installation:

- For CH340 driver installation.

Extract drivers.zip
Install the SETUP.exe



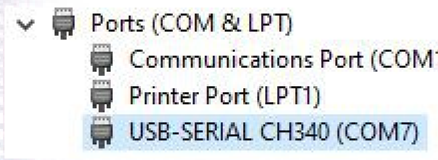
```
..
DRVSETUP64
CH341PT.DLL
CH341S64.SYS
CH341S98.SYS
ch341SER.CAT
CH341SER.INF
CH341SER.SYS
CH341SER.VXD
SETUP.EXE
```

After installation, restart PC (if necessary).

OR

- Plug-in the cable with CH340 driver module

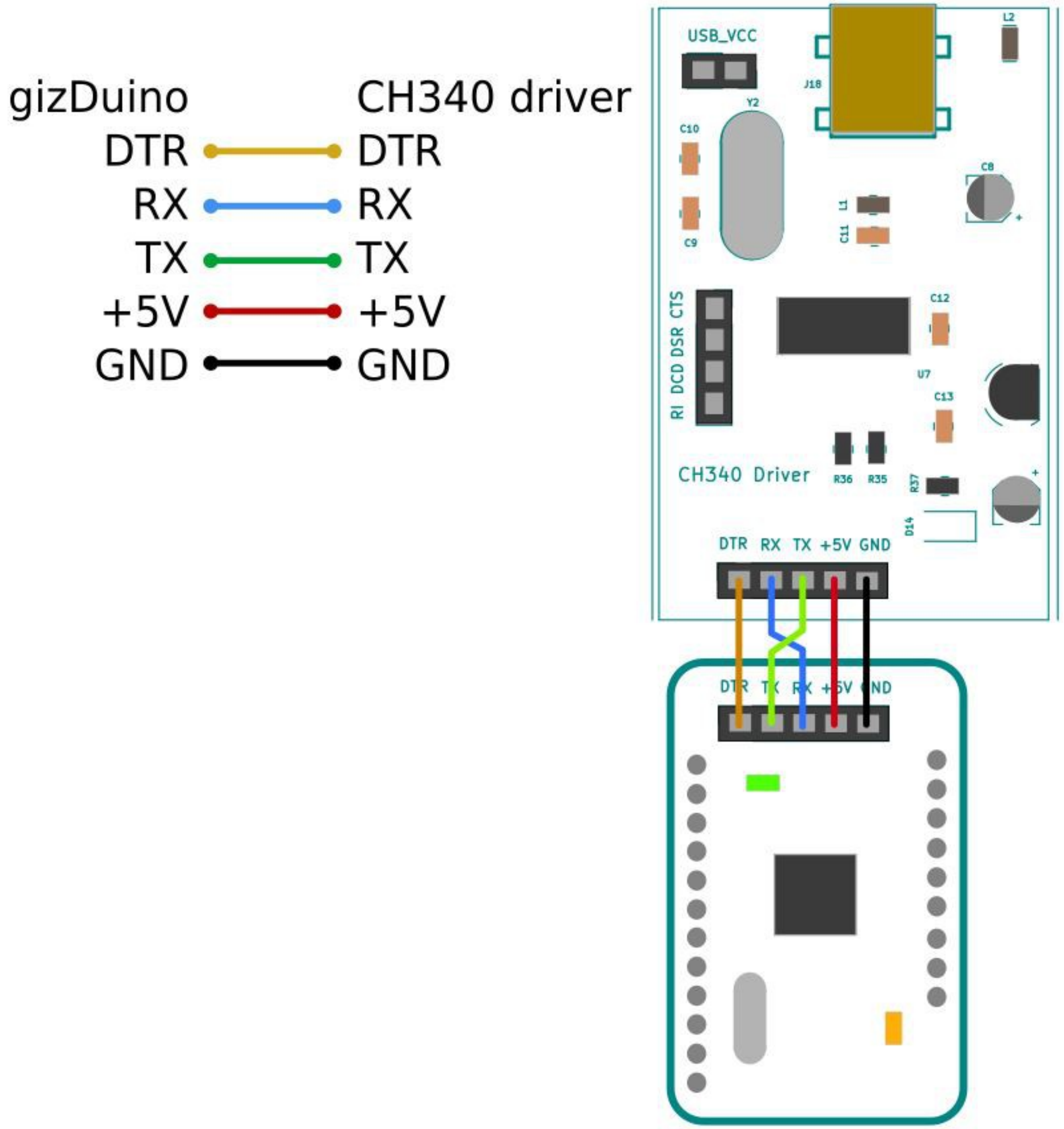
To PC and open the Device Manager> ports>
>USB Serial (Right-Click then Update driver)



```
Ports (COM & LPT)
    Communications Port (COM1)
    Printer Port (LPT1)
    USB-SERIAL CH340 (COM7)
```

← Comport number shown here

CH340 Driver Connections to gizDuino mini328P

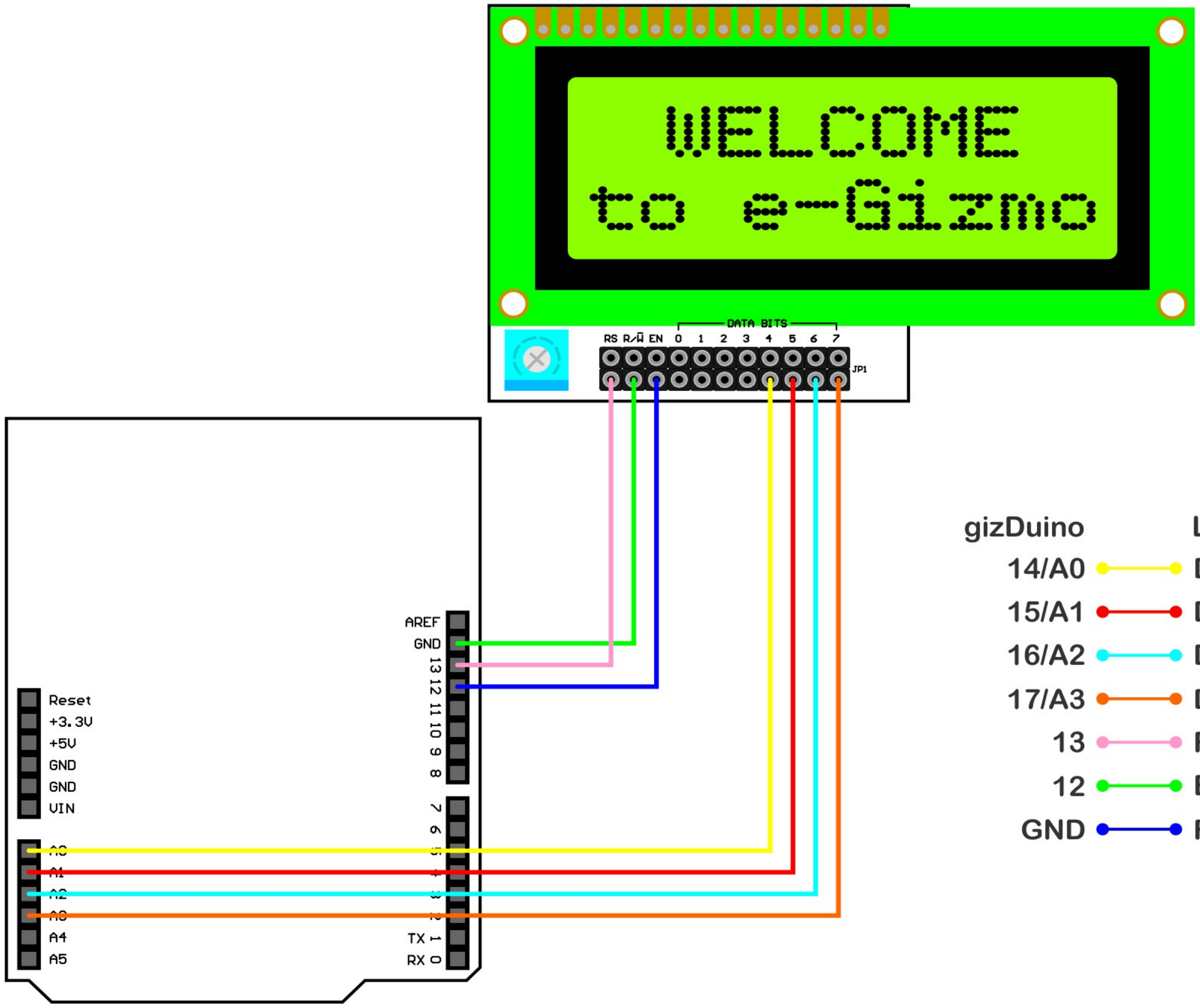


1.Connect the wires.
2.Open the Arduino IDE (modified).
3.Board select: gizDuino mini Atmega328P.
4. Connect the USB cable type mini b - Type A to USB PC port.
5. Select the COM port number.

For example program:
Go to File>Example>Basics>Blink.
Click Upload.

Other board interface
- Arduino Pro mini 168/328 (+5V)
- gizDuino+ Atmega164/324/644
- gizDuino mini 168/328
- devices that has serial connections with RX/TX, DTR

2x16 Character LCD Display



Use to display value,
Data and text message.

Library used:
LiquidCrystal
(how to add library in arduino
- see the next page)

gizDuino		LCD Module
14/A0	🟨	DATA 4
15/A1	🟥	DATA 5
16/A2	🟦	DATA 6
17/A3	🟧	DATA 7
13	🩷	RS
12	🟩	EN
GND	🔵	R/W

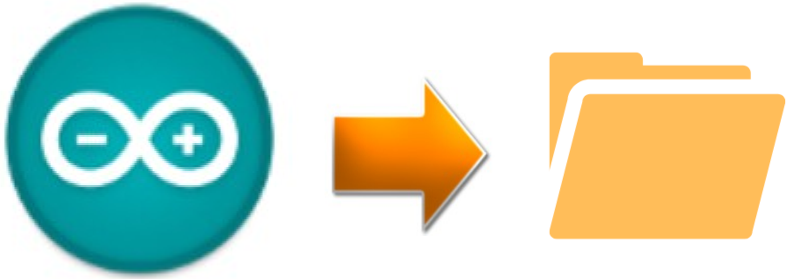
Adding Library to Arduino IDE

There are two ways on how to add library in Arduino IDE but you need to choose one

1. My Documents folder
   - Arduino > libraries > LiquidCrystal folder (which contains: example folder, .h, .cpp, keywords)

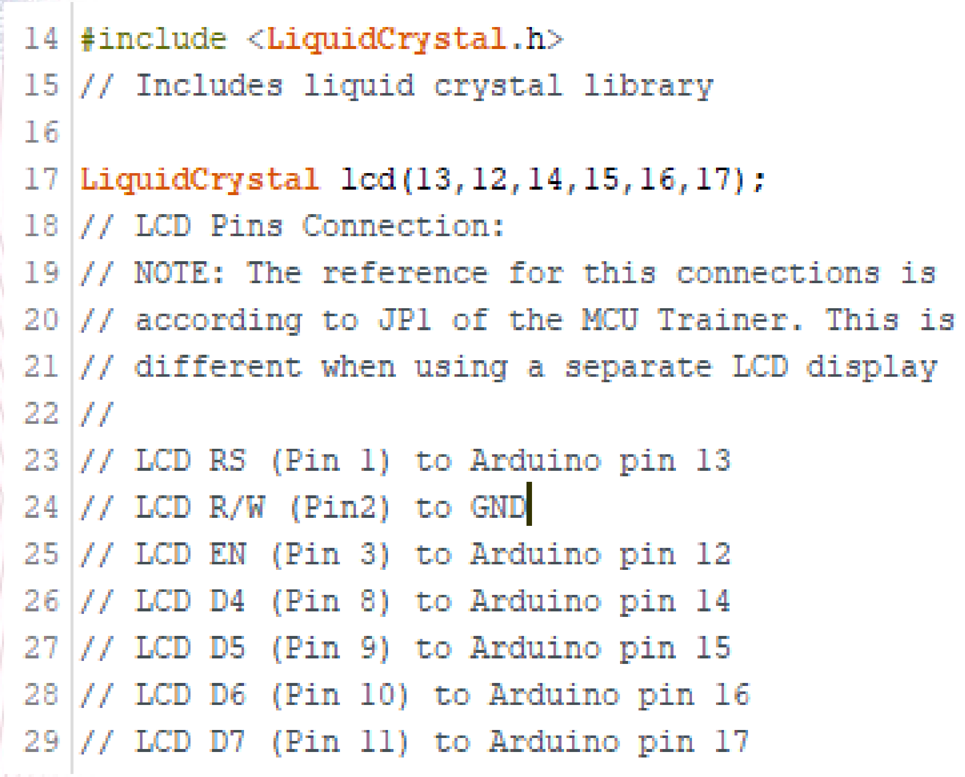
Note: The folder's file name should be the same as the .cpp & .h filename.

2. Arduino IDE 1.8.x folder
   - libraries > LiquidCrystal folder.



Everytime you add/place new library, you must restart your Arduino IDE application.

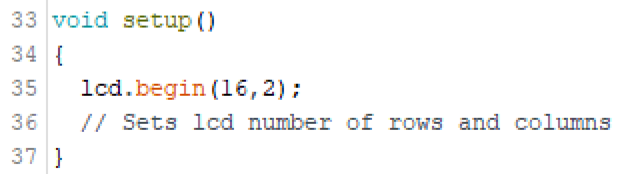
2x16 LCD Display (Library and pin connection)



```
#include <LiquidCrystal.h>
// Includes liquid crystal library

LiquidCrystal lcd(13,12,14,15,16,17);
// LCD Pins Connection:
// NOTE: The reference for this connections is
// according to JP1 of the MCU Trainer. This is
// different when using a separate LCD display
//
// LCD RS (Pin 1) to Arduino pin 13
// LCD R/W (Pin2) to GND
// LCD EN (Pin 3) to Arduino pin 12
// LCD D4 (Pin 8) to Arduino pin 14
// LCD D5 (Pin 9) to Arduino pin 15
// LCD D6 (Pin 10) to Arduino pin 16
// LCD D7 (Pin 11) to Arduino pin 17
```

- Library used

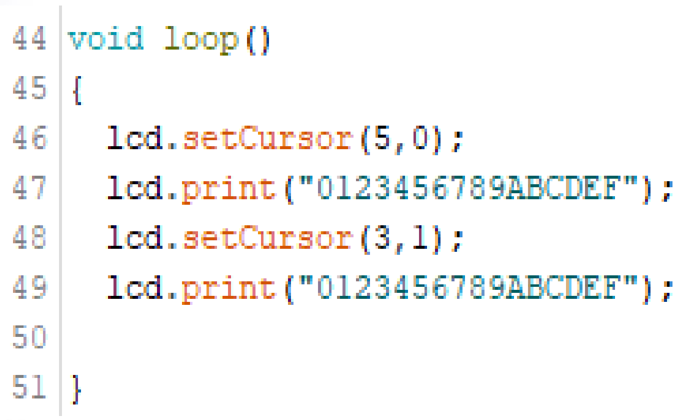
- LCD pins connection



```
void setup()
{
  lcd.begin(16,2);
  // Sets lcd number of rows and columns
}
```

- Setup
Lcd begin set to 16 x 2
16 number of columns
2 number of rows

2x16 LCD Display (loop)

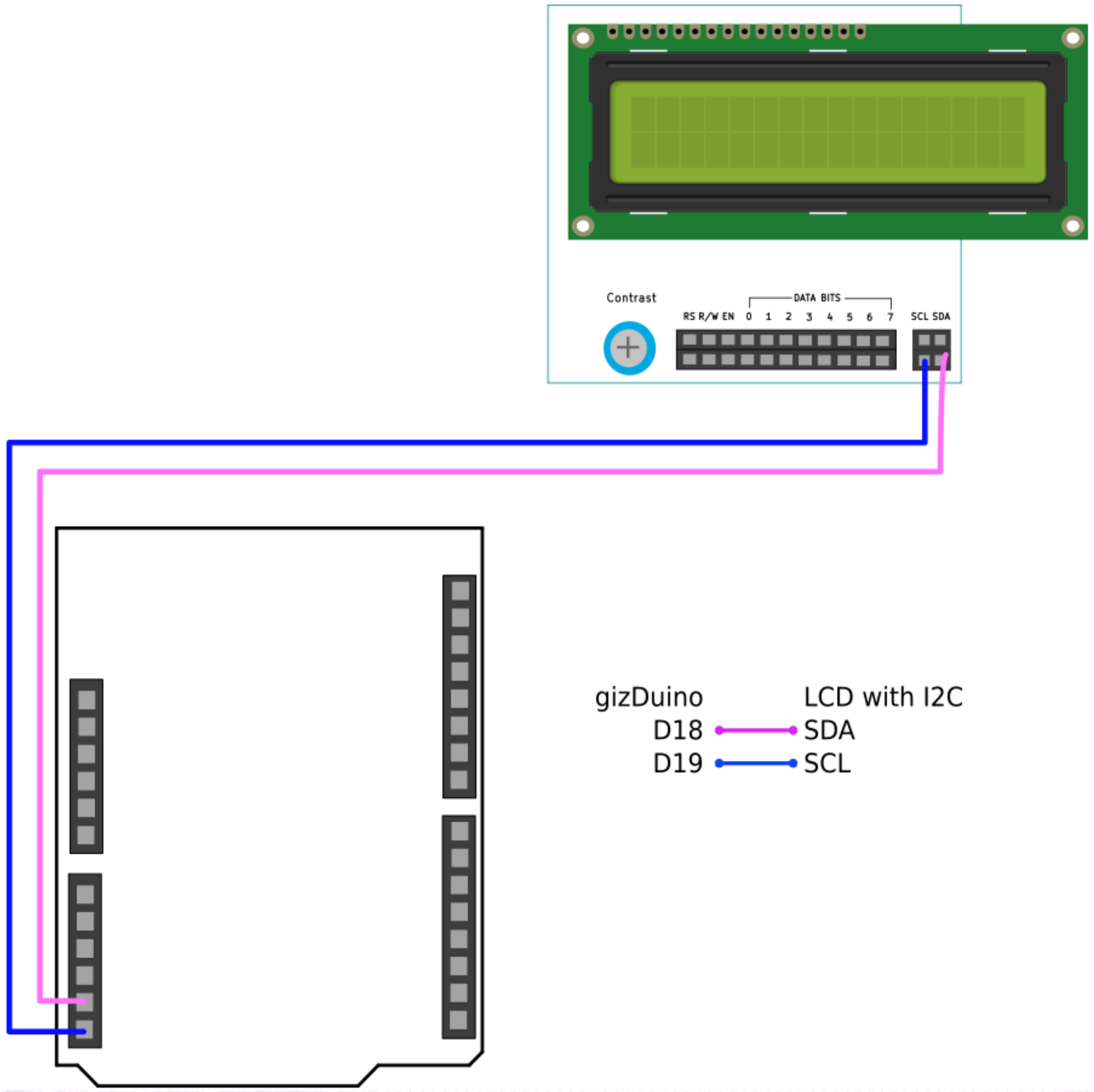


```
void loop()
{
  lcd.setCursor(5,0);
  lcd.print("0123456789ABCDEF");
  lcd.setCursor(3,1);
  lcd.print("0123456789ABCDEF");

}
```

- loop

- set cursor to column 5 and row 0
  - print string to lcd
- set cursor again to column 3 and row 1
  - print string

2x16 Character LCD Display with I2C

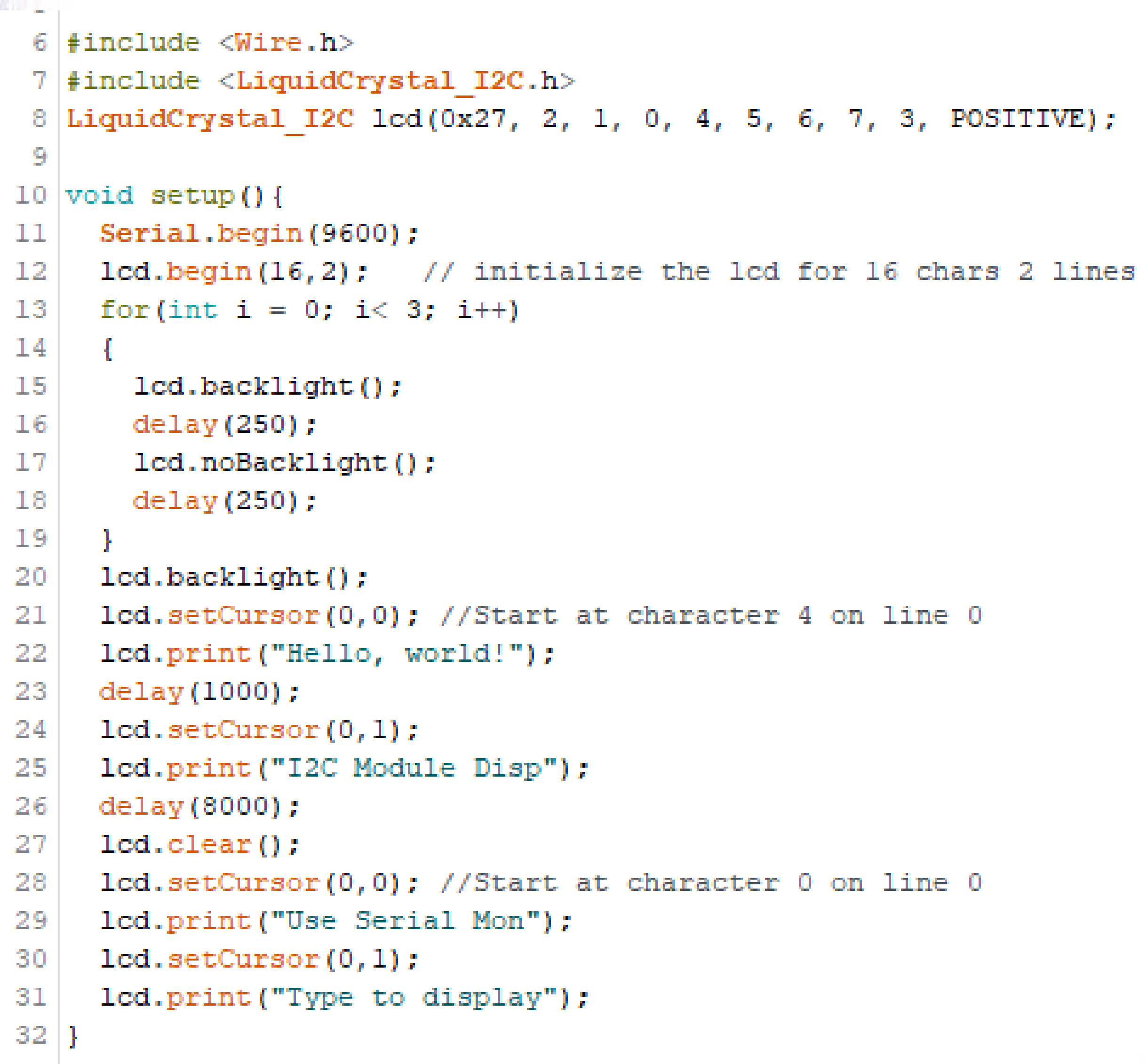


gizDuino	LCD with I2C
D18	SDA
D19	SCL

For LCD with I2C, Attached the LCD to I2C module slot.

Connect the GizDuino to LCDI2C
SDA/D18 → SDA
SCL/D19 → SCL

LCDI2C Sketch (setup)



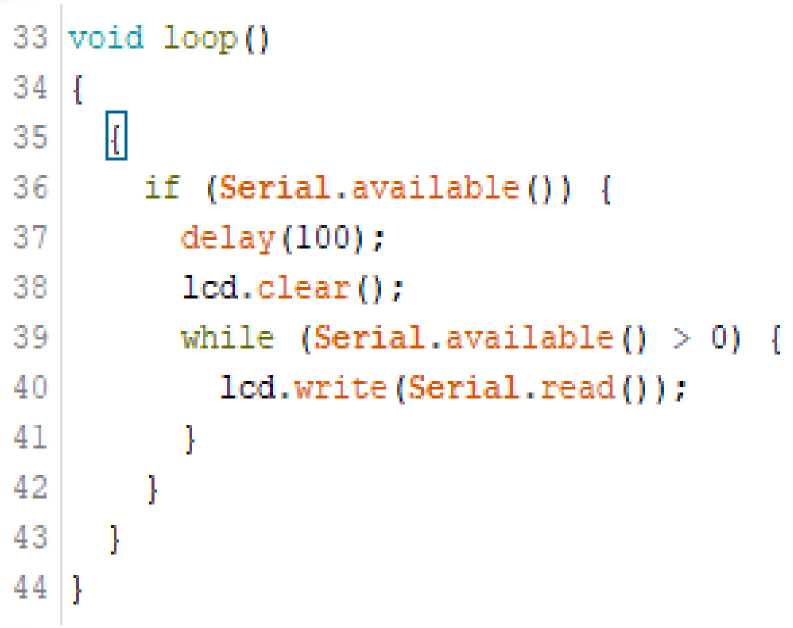
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup(){
  Serial.begin(9600);
  lcd.begin(16,2);   // initialize the lcd for 16 chars 2 lines
  for(int i = 0; i< 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();
  lcd.setCursor(0,0); //Start at character 4 on line 0
  lcd.print("Hello, world!");
  delay(1000);
  lcd.setCursor(0,1);
  lcd.print("I2C Module Disp");
  delay(8000);
  lcd.clear();
  lcd.setCursor(0,0); //Start at character 0 on line 0
  lcd.print("Use Serial Mon");
  lcd.setCursor(0,1);
  lcd.print("Type to display");
}
```

- Library used
  Wire
  LiquidCrystal_I2C

-setup
  Set baudrate to 9600
- lcd.begin(16,2); // if 2x16 LCD
- for loop, for blinking the backlight
  3X times on/off

- turn on backlight
- set cursor to origin
- print string to lcd

LCDI2C Sketch (loop)

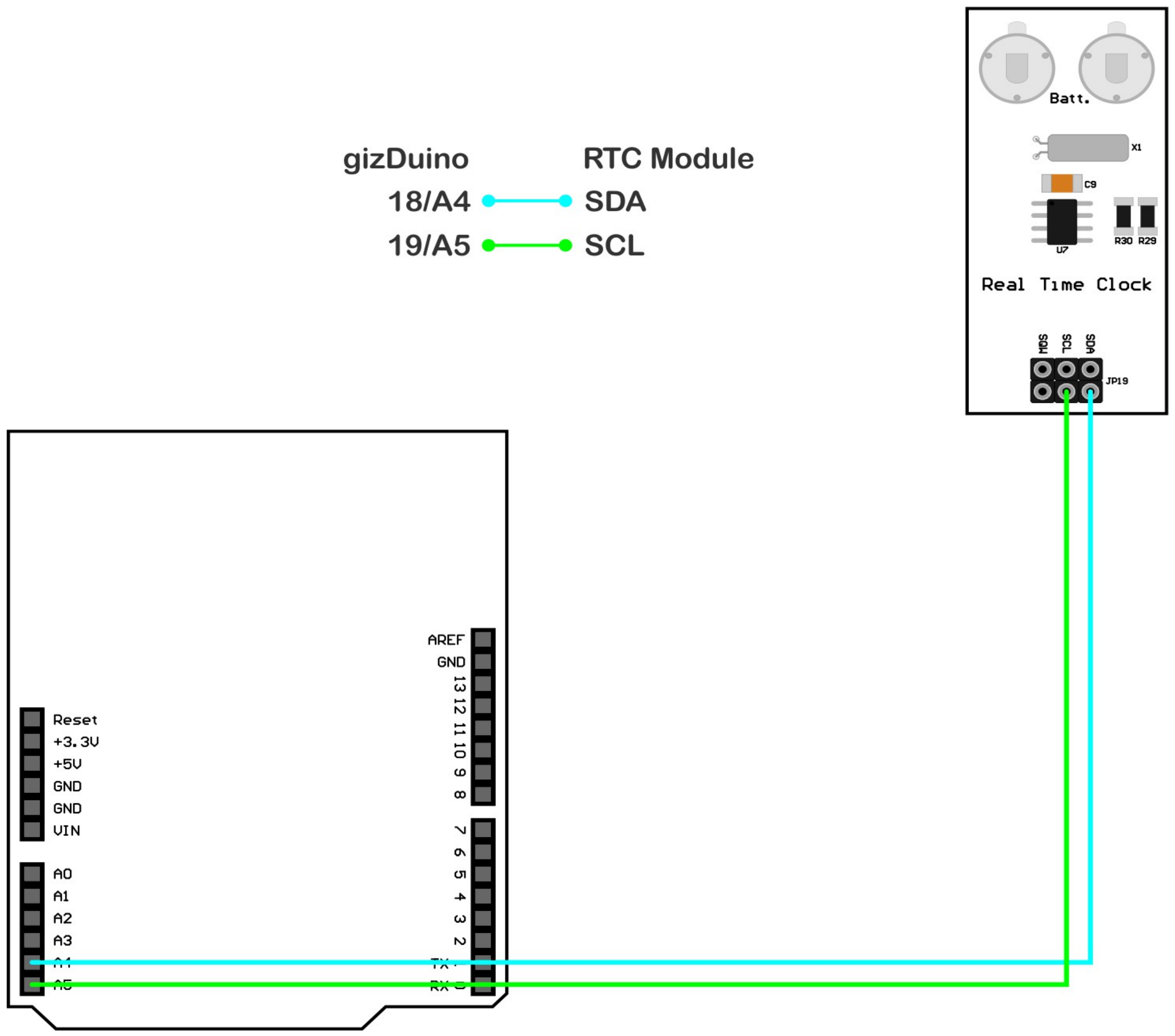


```
void loop()
{
  {
    if (Serial.available()) {
      delay(100);
      lcd.clear();
      while (Serial.available() > 0) {
        lcd.write(Serial.read());
      }
    }
  }
}
```

- using serial available
  We can get the data from the
Serial Monitor after sending
It will display/print on the LCD.

Real Time Clock DS1307

gizDuino	RTC Module
18/A4	SDA
19/A5	SCL

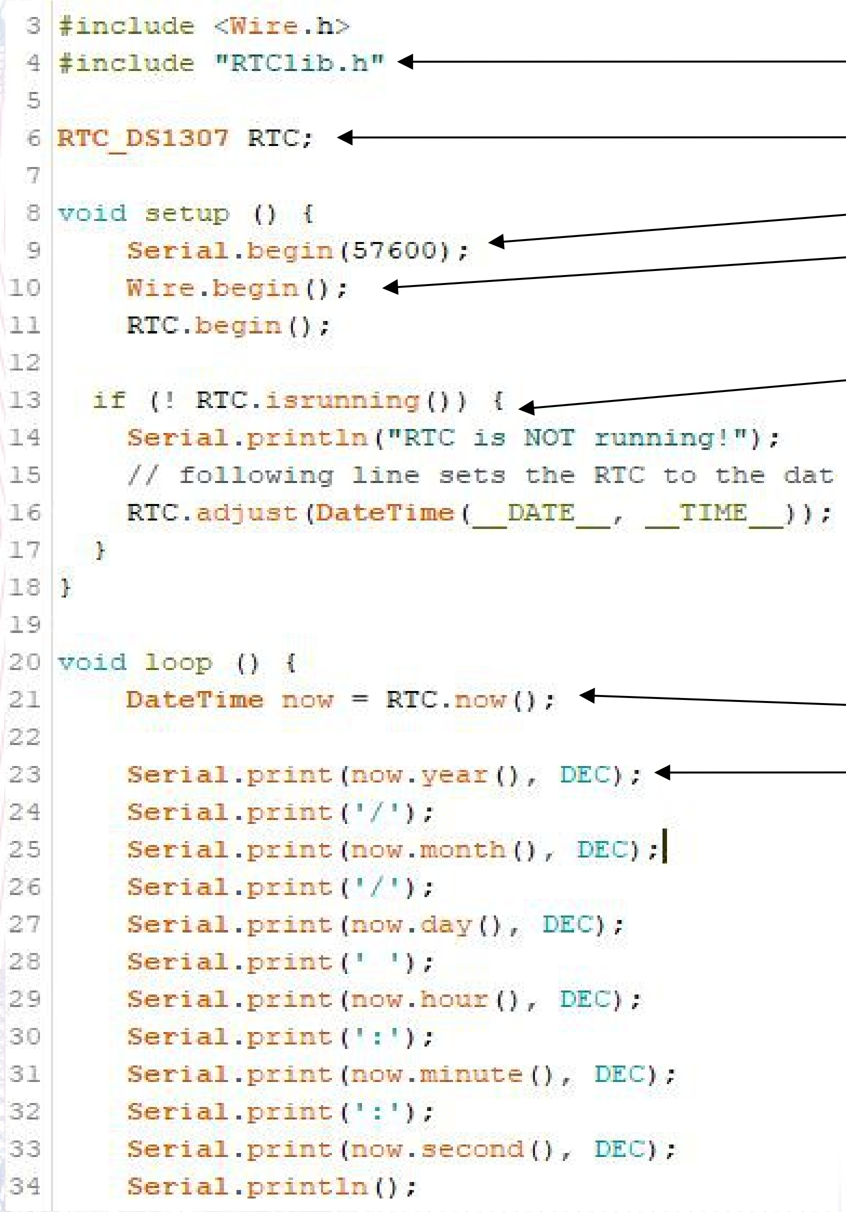


Looking for real time data?

This RTC module has a Real time value of date and time for project such as RFID attendance, Library Login/out, Monitoring and Database via i2c serial Commnunication of module.

Displays data to serial connections to the pc terminal, Or to lcd display through Parallel connections.

DS1307RTC Sample sketch



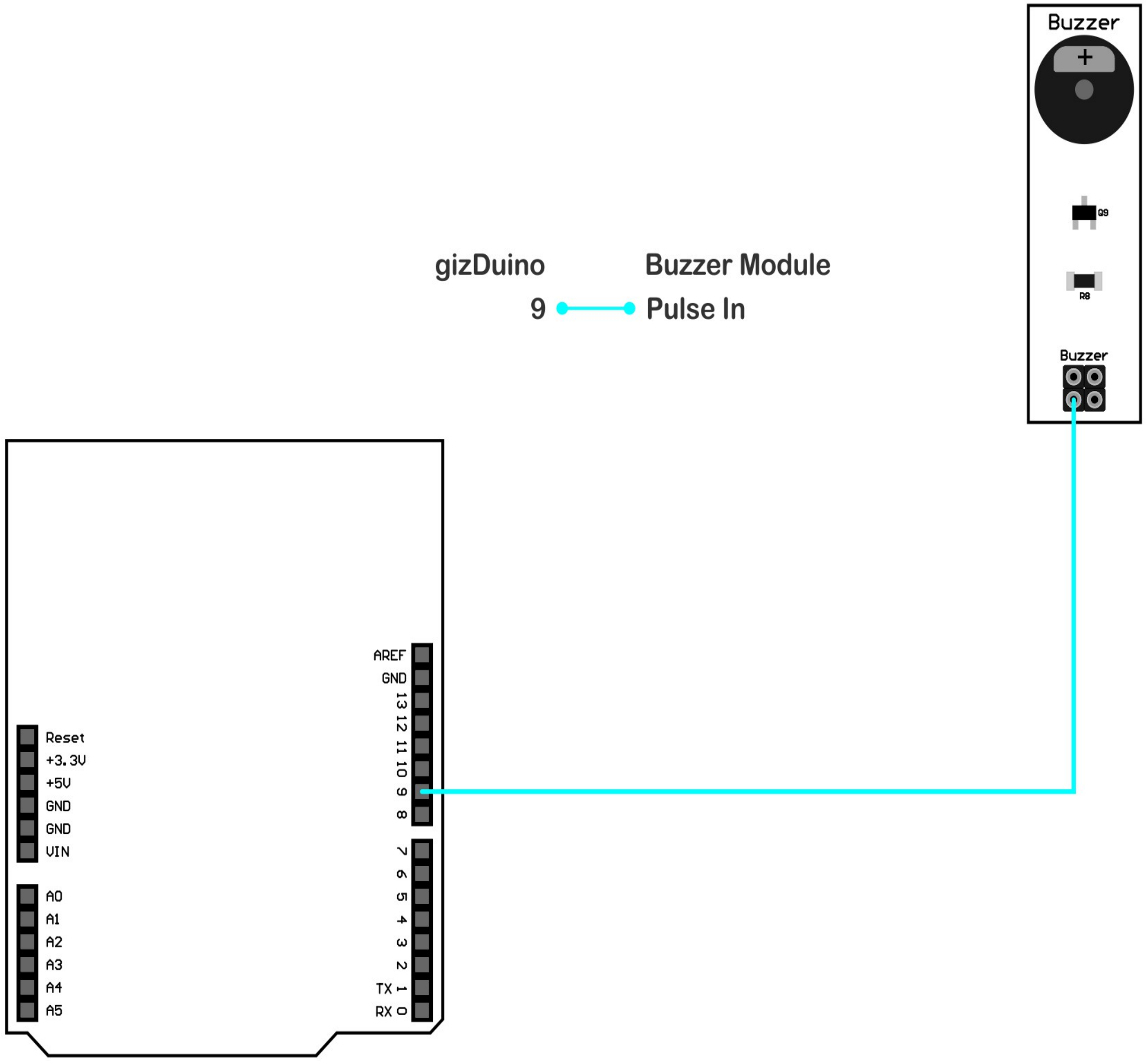
```
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
    Serial.begin(57600);
    Wire.begin();
    RTC.begin();

  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the dat
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
    DateTime now = RTC.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
```

- Library
- name it your RTC
- Set baudrate 57600
- start libraries with begin functions

- indication function, if RTC module is not running.
- RTC.adjust(DateTime)

- set RTC.now for real time data
- use Serial.print() to display in the terminal
- date settings use:
  - now.year(), now.month(),now.day()
  - set to DEC or decimal
- time settings use:
  - now.hour(),now.minute().now.second()

Passive Buzzer

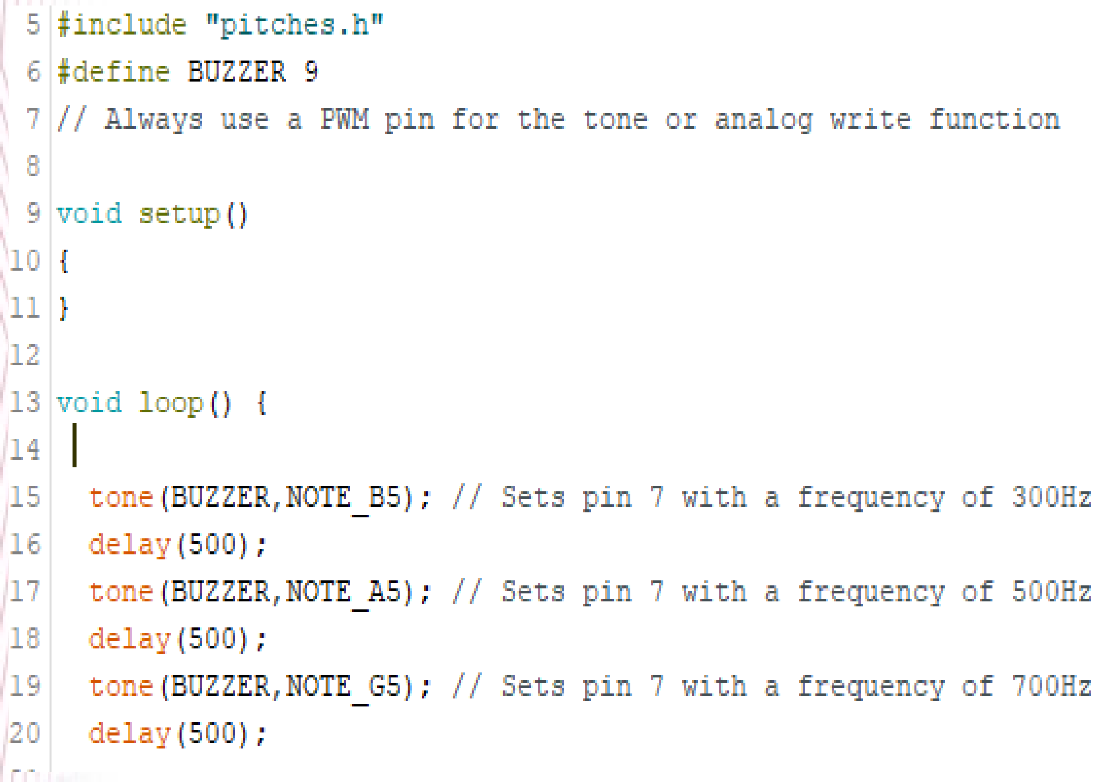


In Passive buzzer
this in not an ordinary buzzer where you can put supply on it. It is specialize for receiving frequency from gizDuino PWM Pins 3,5,6,9,10, or 11.

The frequency ranges from 31 to 4.9Khz (see the pitches.h).

Passive Buzzer Sketch tone_1

- Library
- set variable name 'BUZZER' to digital pin 9

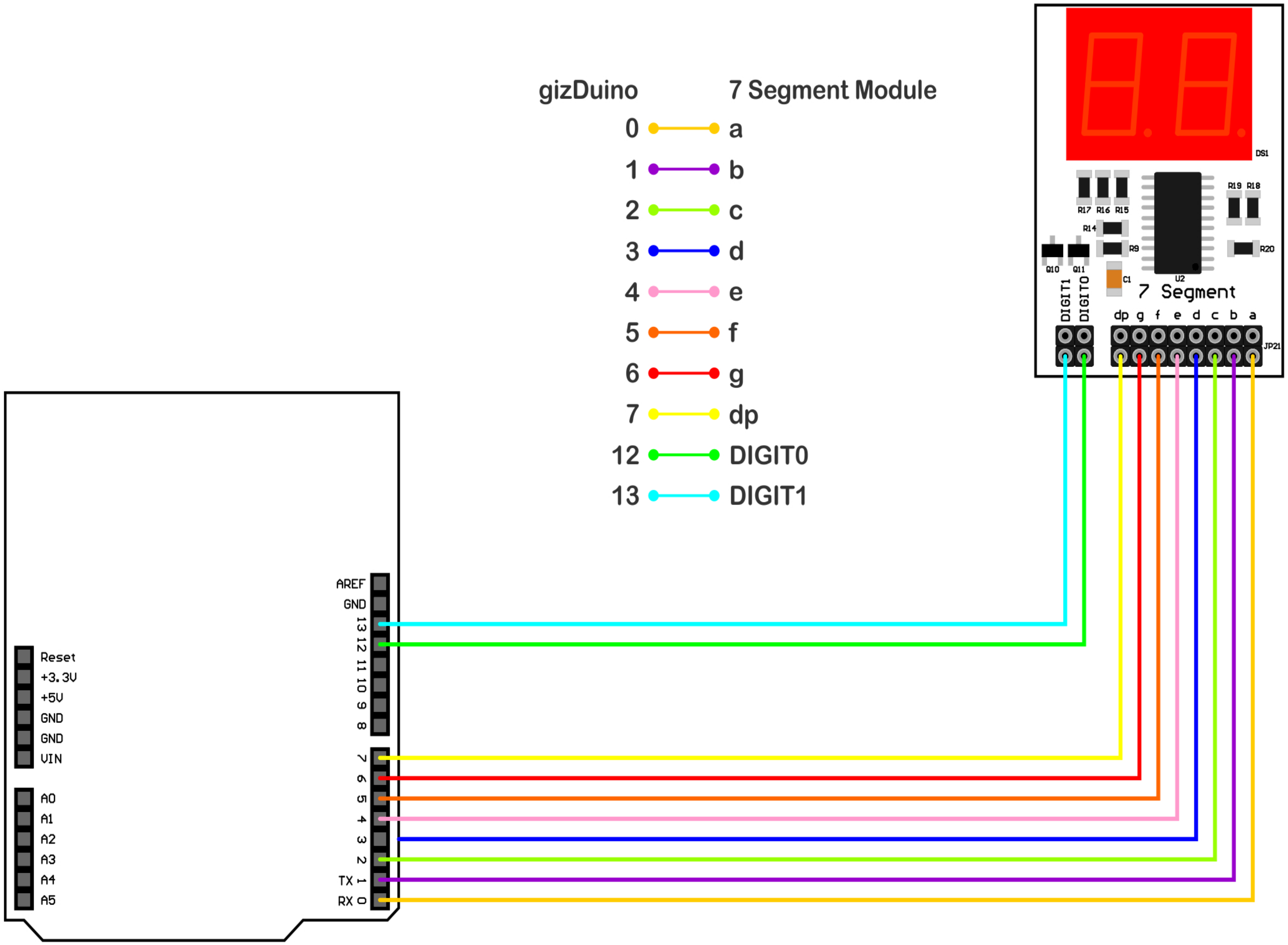


```
#include "pitches.h"
#define BUZZER 9
// Always use a PWM pin for the tone or analog write function

void setup()
{
}

void loop() {

  tone(BUZZER,NOTE_B5); // Sets pin 7 with a frequency of 300Hz
  delay(500);
  tone(BUZZER,NOTE_A5); // Sets pin 7 with a frequency of 500Hz
  delay(500);
  tone(BUZZER,NOTE_G5); // Sets pin 7 with a frequency of 700Hz
  delay(500);
```

tone (pin number, frequency);
or frequency see the pithces.h

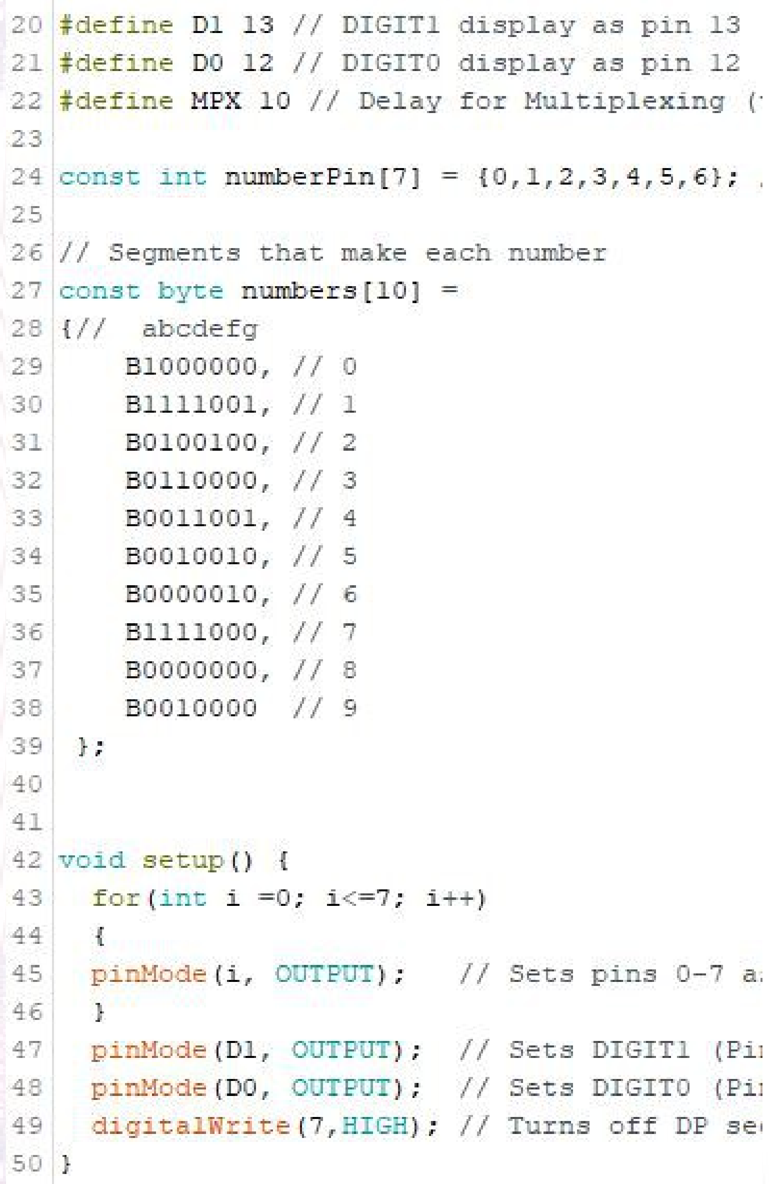
2 digits 7 segment Display

gizDuino	7 Segment Module
0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	dp
12	DIGIT0
13	DIGIT1



We have here 2 digits 7 segment display. For counter and Countdown display.

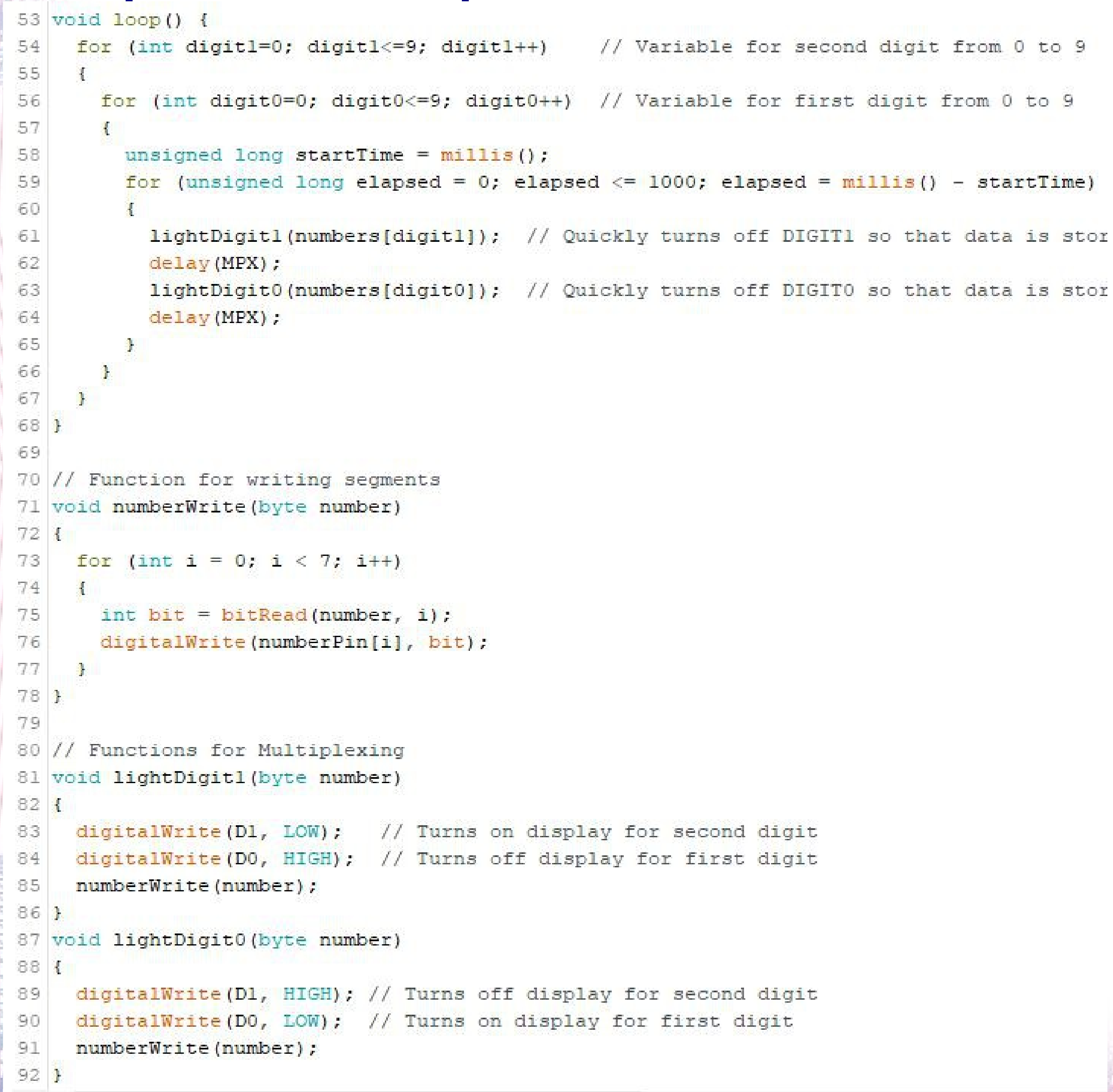
Sample sketch pin assign and setup



```
#define D1 13 // DIGIT1 display as pin 13
#define D0 12 // DIGIT0 display as pin 12
#define MPX 10 // Delay for Multiplexing (

const int numberPin[7] = {0,1,2,3,4,5,6};

// Segments that make each number
const byte numbers[10] =
{//  abcdefg
    B1000000, // 0
    B1111001, // 1
    B0100100, // 2
    B0110000, // 3
    B0011001, // 4
    B0010010, // 5
    B0000010, // 6
    B1111000, // 7
    B0000000, // 8
    B0010000  // 9
 };


void setup() {
  for(int i =0; i<=7; i++)
  {
  pinMode(i, OUTPUT);   // Sets pins 0-7 a
  }
  pinMode(D1, OUTPUT);  // Sets DIGIT1 (Pi
  pinMode(D0, OUTPUT);  // Sets DIGIT0 (Pi
  digitalWrite(7,HIGH); // Turns off DP se
}
```

- pins assignment
- binary number equivalent to 0-9
- setup
- setting up all the pins assignment to output

Sample sketch loop



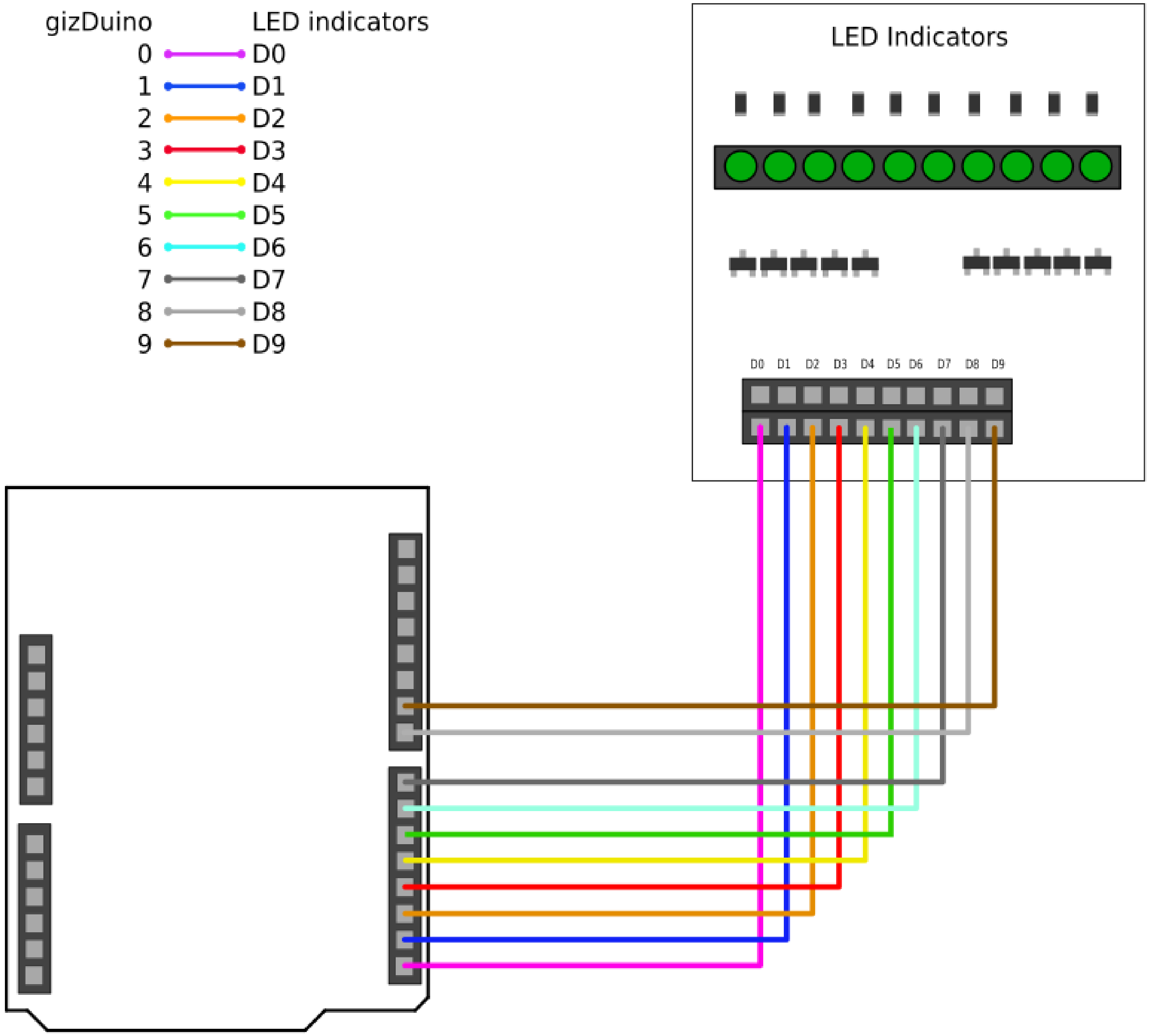
```
void loop() {
  for (int digit1=0; digit1<=9; digit1++)     // Variable for second digit from 0 to 9
  {
    for (int digit0=0; digit0<=9; digit0++)  // Variable for first digit from 0 to 9
    {
      unsigned long startTime = millis();
      for (unsigned long elapsed = 0; elapsed <= 1000; elapsed = millis() - startTime)
      {
        lightDigit1(numbers[digit1]);  // Quickly turns off DIGIT1 so that data is stor
        delay(MPX);
        lightDigit0(numbers[digit0]);  // Quickly turns off DIGIT0 so that data is stor
        delay(MPX);
      }
    }
  }
}

// Function for writing segments
void numberWrite(byte number)
{
  for (int i = 0; i < 7; i++)
  {
    int bit = bitRead(number, i);
    digitalWrite(numberPin[i], bit);
  }
}

// Functions for Multiplexing
void lightDigit1(byte number)
{
  digitalWrite(D1, LOW);   // Turns on display for second digit
  digitalWrite(D0, HIGH);  // Turns off display for first digit
  numberWrite(number);
}
void lightDigit0(byte number)
{
  digitalWrite(D1, HIGH); // Turns off display for second digit
  digitalWrite(D0, LOW);  // Turns on display for first digit
  numberWrite(number);
}
```

- loop
- for loop function for
  Second & first digit

- function
  For writing segments
- bitRead

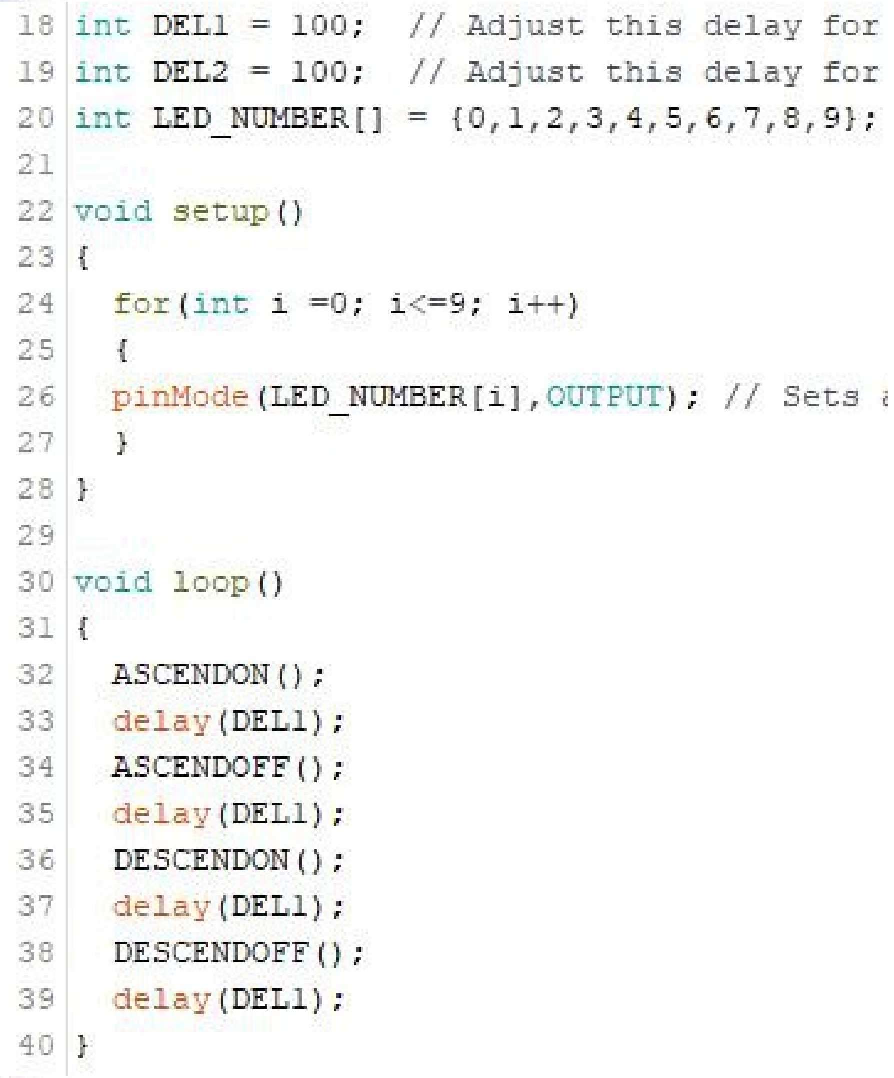
LED Indicators

gizDuino	LED indicators
0	D0
1	D1
2	D2
3	D3
4	D4
5	D5
6	D6
7	D7
8	D8
9	D9



The 10 LED indicator use
To practice on how to apply
The LED to other devices.
Like Running light, on/off
Sequence etc.

Sample sketch for 10 leds



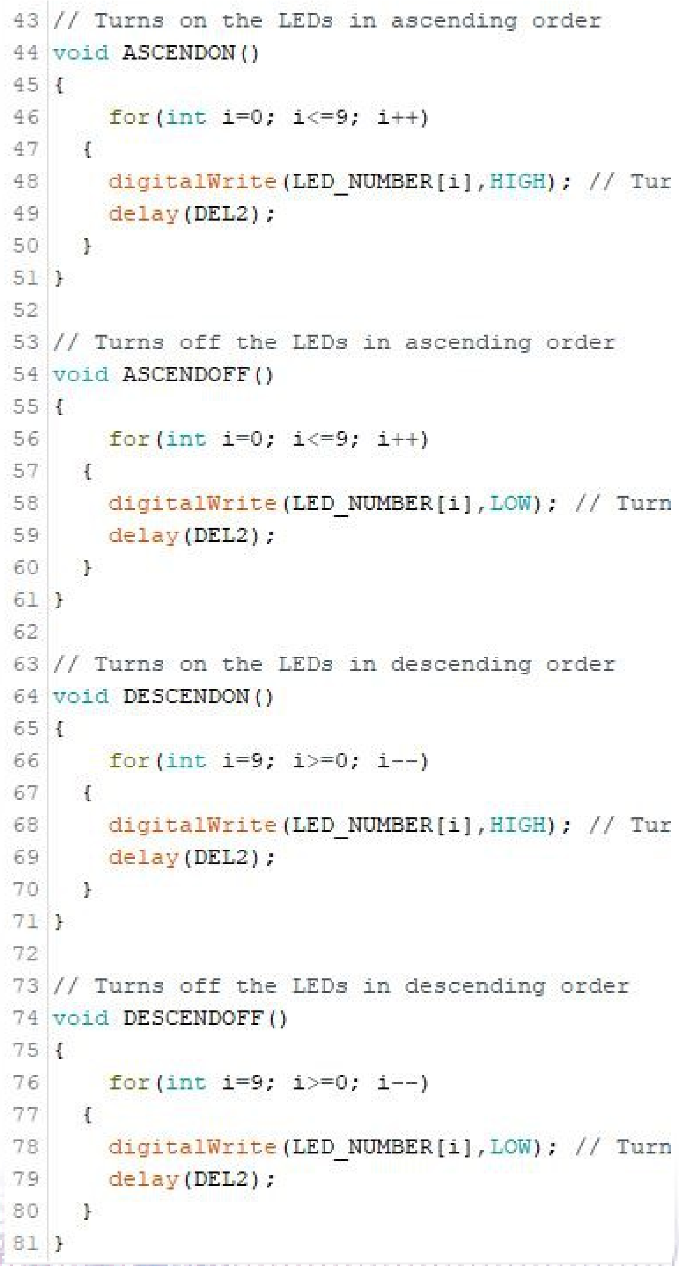
```
int DEL1 = 100;   // Adjust this delay for
int DEL2 = 100;   // Adjust this delay for
int LED_NUMBER[] = {0,1,2,3,4,5,6,7,8,9};

void setup()
{
  for(int i =0; i<=9; i++)
  {
  pinMode(LED_NUMBER[i],OUTPUT); // Sets
  }
}

void loop()
{
  ASCENDON();
  delay(DEL1);
  ASCENDOFF();
  delay(DEL1);
  DESCENDON();
  delay(DEL1);
  DESCENDOFF();
  delay(DEL1);
}
```

- delays
- LED array 0-9

- setup
- for function setting up
  the 0-9 digital pins to output

- loop
- created functions inserted
For ascending ON/OFF
And descending ON/OFF.

Sample sketch on how to make functions



```
// Turns on the LEDs in ascending order
void ASCENDON()
{
    for(int i=0; i<=9; i++)
  {
    digitalWrite(LED_NUMBER[i],HIGH); // Tur
    delay(DEL2);
  }
}

// Turns off the LEDs in ascending order
void ASCENDOFF()
{
    for(int i=0; i<=9; i++)
  {
    digitalWrite(LED_NUMBER[i],LOW); // Turn
    delay(DEL2);
  }
}

// Turns on the LEDs in descending order
void DESCENDON()
{
    for(int i=9; i>=0; i--)
  {
    digitalWrite(LED_NUMBER[i],HIGH); // Tur
    delay(DEL2);
  }
}

// Turns off the LEDs in descending order
void DESCENDOFF()
{
    for(int i=9; i>=0; i--)
  {
    digitalWrite(LED_NUMBER[i],LOW); // Turn
    delay(DEL2);
  }
}
```

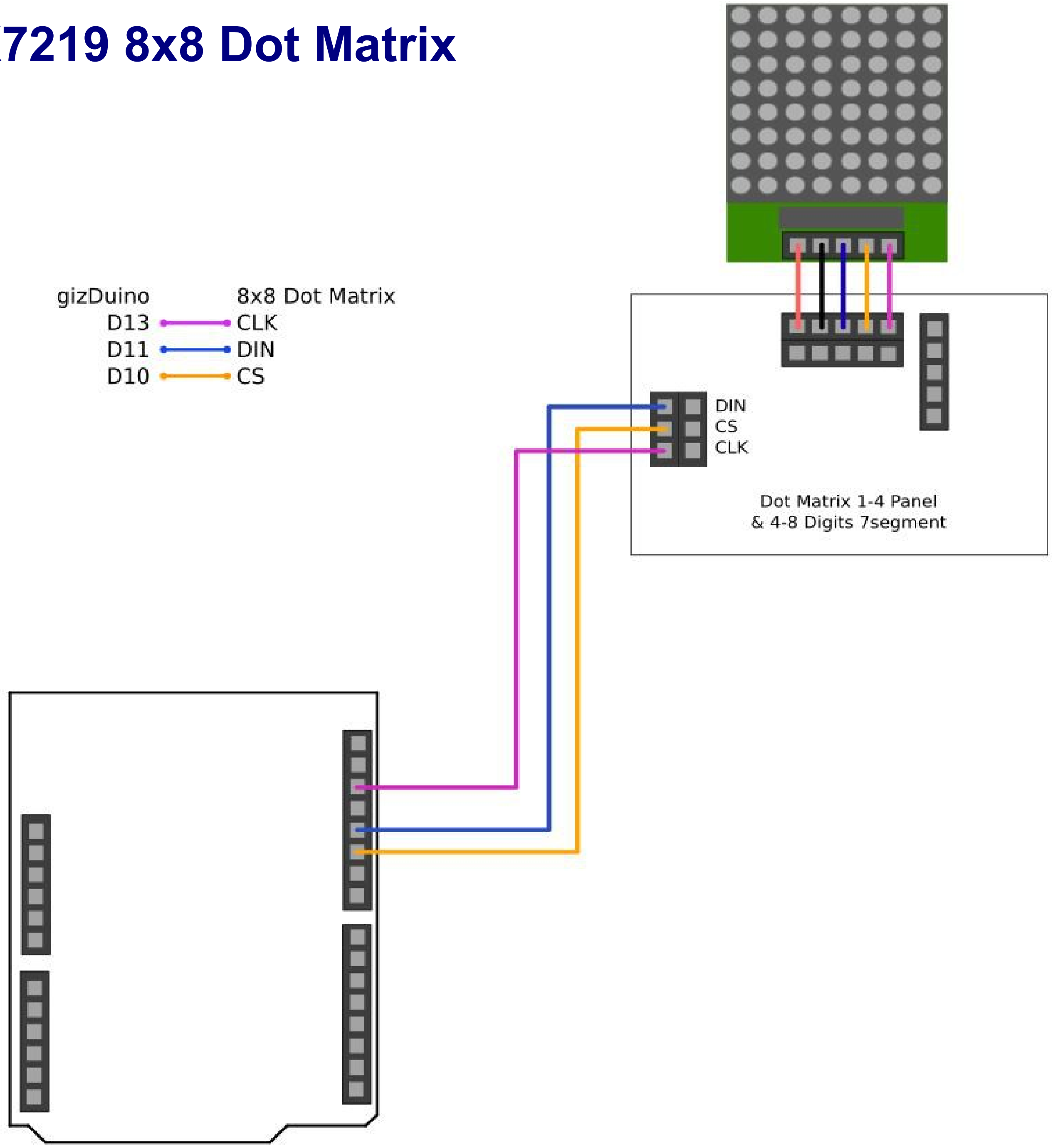
- Ascending
- using for loop function

from 0-9 ON and ascending OFF.

- Decending

From 9-0 ON and decending OFF

MAX7219 8x8 Dot Matrix

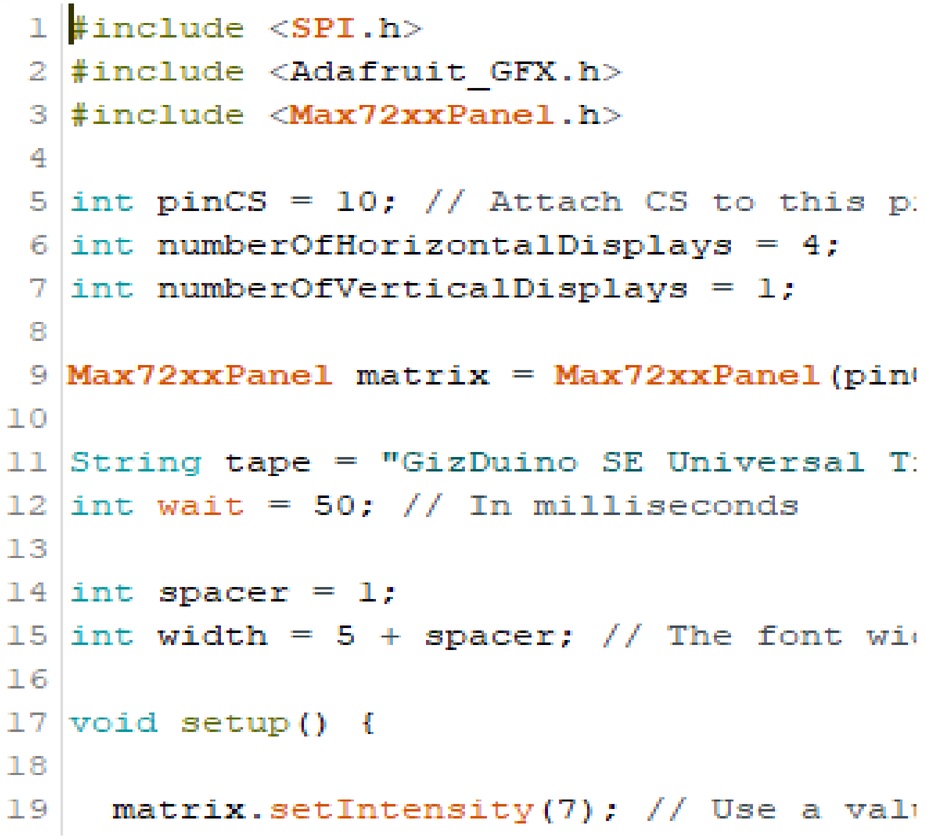


Attached the 8x8 Dot Matrix module and Connect the correct pins For DIN, CS, CLK

This module can display Icons, emoticons and text message (scrolling only) With about 50ms delay for Readable speed.

Library used: Max72xxPanel

Sample sketch for 8x8 dot matrix using MAX7219 (library,setup)



```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10; // Attach CS to this p
int numberOfHorizontalDisplays = 4;
int numberOfVerticalDisplays = 1;

Max72xxPanel matrix = Max72xxPanel(pin

String tape = "GizDuino SE Universal T
int wait = 50; // In milliseconds

int spacer = 1;
int width = 5 + spacer; // The font wi

void setup() {

  matrix.setIntensity(7); // Use a val
```

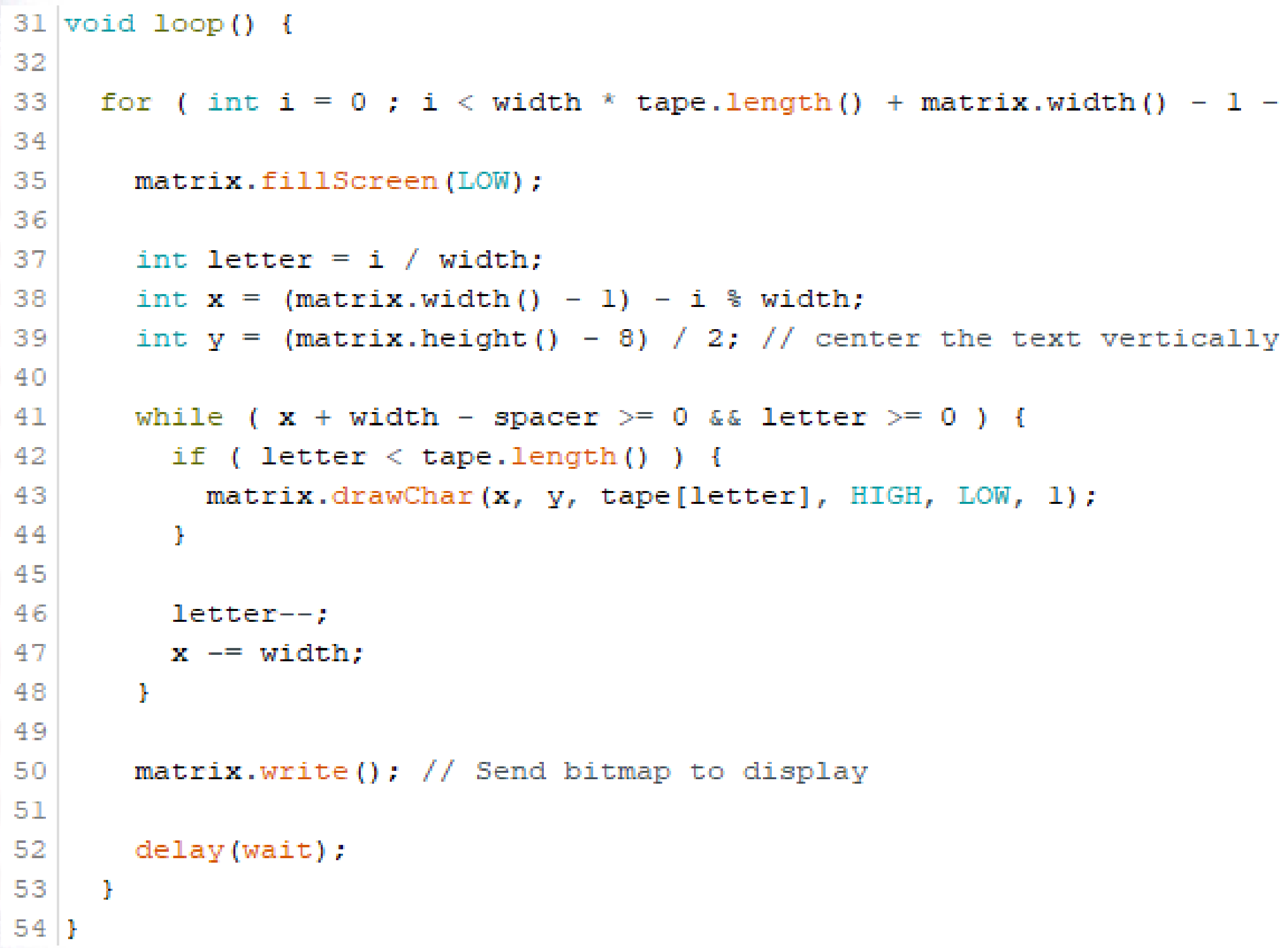
- Library used:
  SPI, Adafruit_GFX,
  MAX72xxPanel

- assign CS pin to 10
- set the number of display

- Type the display message
- wait (delay in ms)

- In Setup
  setIntensity

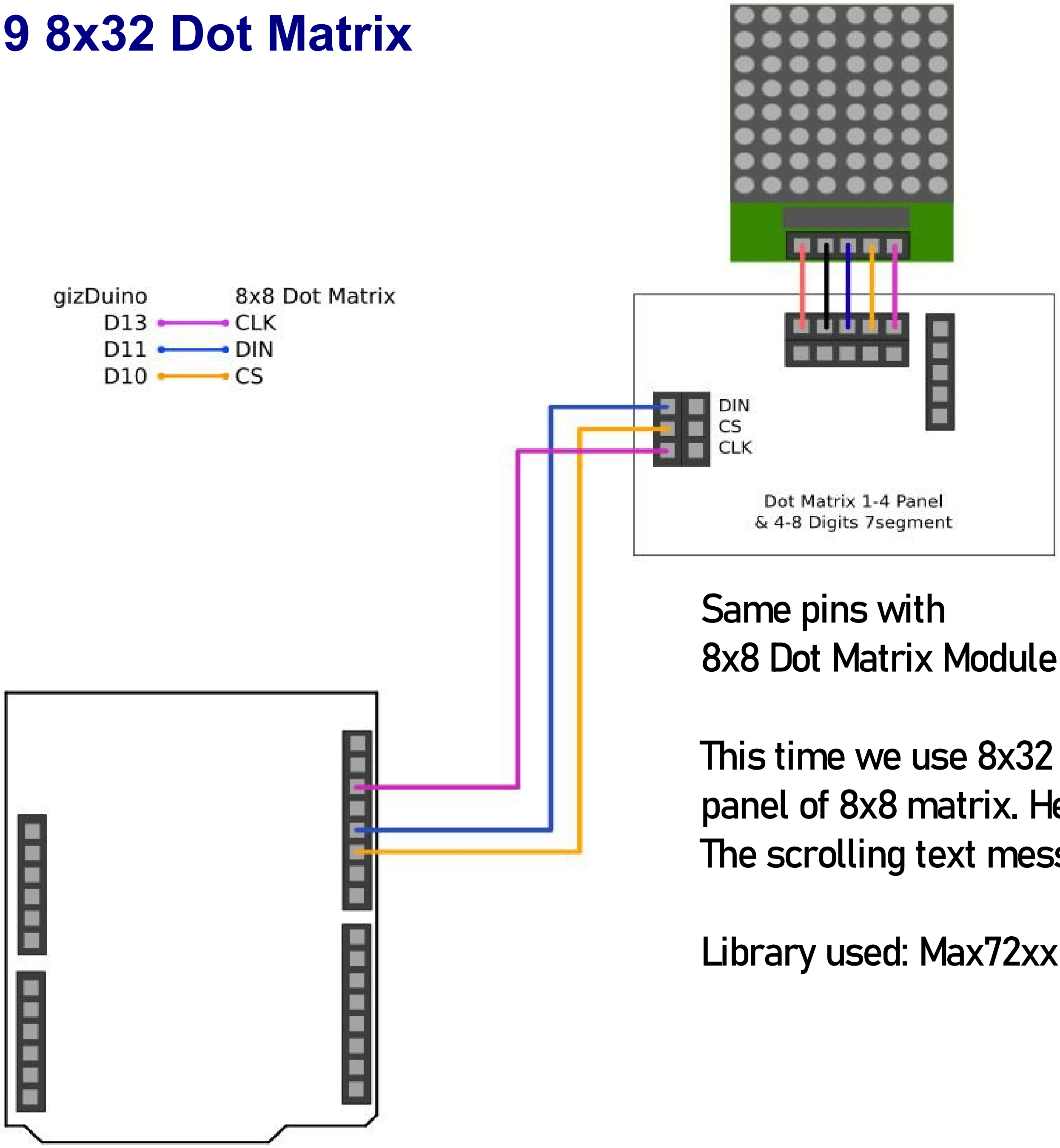
Sample sketch for 8x8 dot matrix using MAX7219 (loop)



```
void loop() {

  for ( int i = 0 ; i < width * tape.length() + matrix.width() - 1 - spacer; i++ ) {

    matrix.fillScreen(LOW);

    int letter = i / width;
    int x = (matrix.width() - 1) - i % width;
    int y = (matrix.height() - 8) / 2; // center the text vertically

    while ( x + width - spacer >= 0 && letter >= 0 ) {
      if ( letter < tape.length() ) {
        matrix.drawChar(x, y, tape[letter], HIGH, LOW, 1);
      }

      letter--;
      x -= width;
    }

    matrix.write(); // Send bitmap to display

    delay(wait);
  }
}
```

- In loop
  Code for scrolling the message display

MAX7219 8x32 Dot Matrix

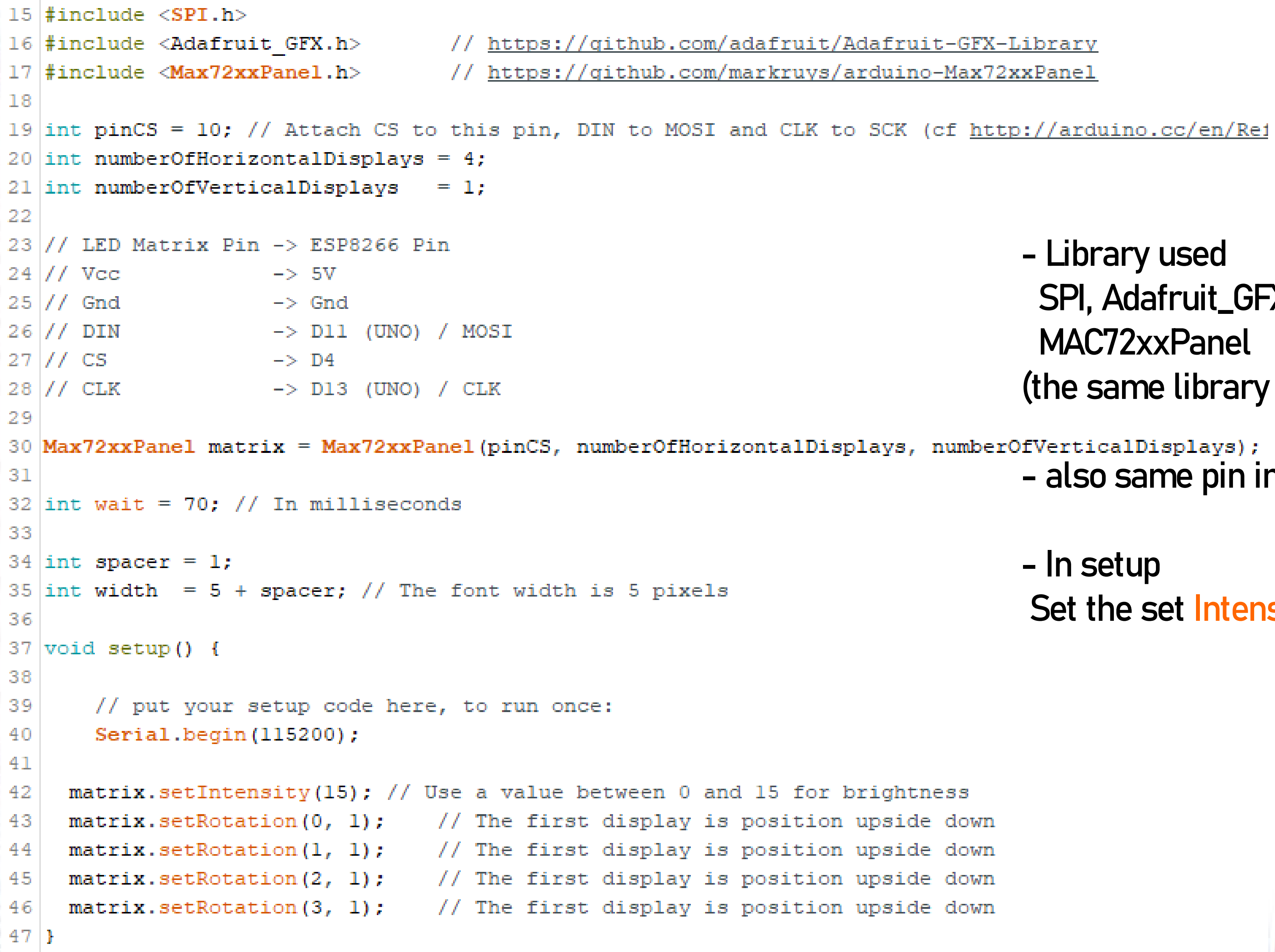


Same pins with
8x8 Dot Matrix Module

This time we use 8x32 dot matrix it has 4
panel of 8x8 matrix. Here you can read more
The scrolling text message.

Library used: Max72xxPanel

Sample sketch for 8x32 dot matrix using MAX7219 (setup)



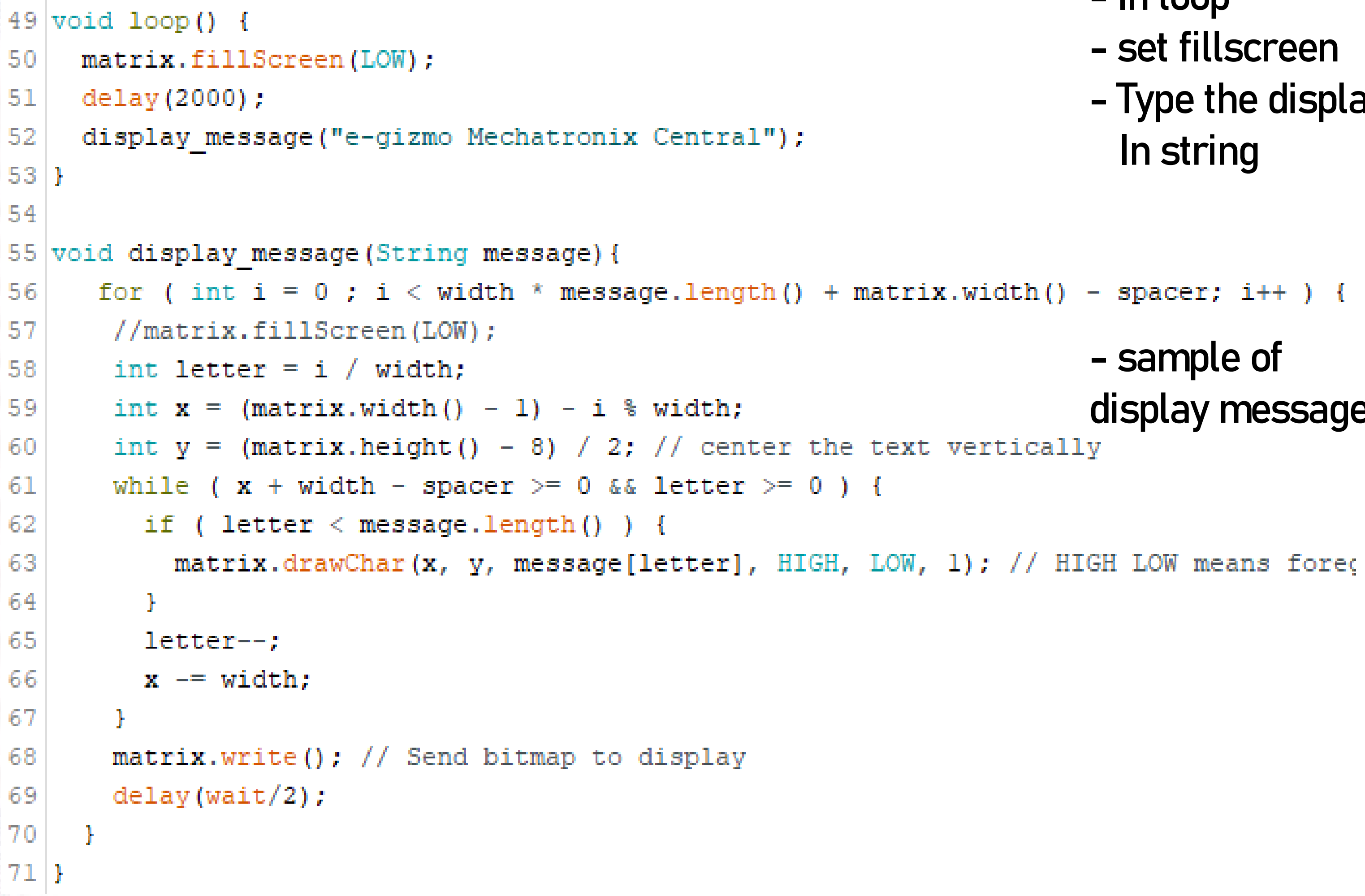
```
#include <SPI.h>
#include <Adafruit_GFX.h>       // https://github.com/adafruit/Adafruit-GFX-Library
#include <Max72xxPanel.h>       // https://github.com/markruys/arduino-Max72xxPanel

int pinCS = 10; // Attach CS to this pin, DIN to MOSI and CLK to SCK (cf http://arduino.cc/en/Re
int numberOfHorizontalDisplays = 4;
int numberOfVerticalDisplays   = 1;

// LED Matrix Pin -> ESP8266 Pin
// Vcc            -> 5V
// Gnd            -> Gnd
// DIN            -> D11 (UNO) / MOSI
// CS             -> D4
// CLK            -> D13 (UNO) / CLK

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

int wait = 70; // In milliseconds

int spacer = 1;
int width  = 5 + spacer; // The font width is 5 pixels

void setup() {

    // put your setup code here, to run once:
    Serial.begin(115200);

  matrix.setIntensity(15); // Use a value between 0 and 15 for brightness
  matrix.setRotation(0, 1);    // The first display is position upside down
  matrix.setRotation(1, 1);    // The first display is position upside down
  matrix.setRotation(2, 1);    // The first display is position upside down
  matrix.setRotation(3, 1);    // The first display is position upside down
}
```

- Library used
  SPI, Adafruit_GFX,
  MAC72xxPanel
  (the same library in 8x8)

- also same pin in 8x8

- In setup
  Set the set Intensity, Rotation

Sample sketch for 8x32 dot matrix using MAX7219 (loop)



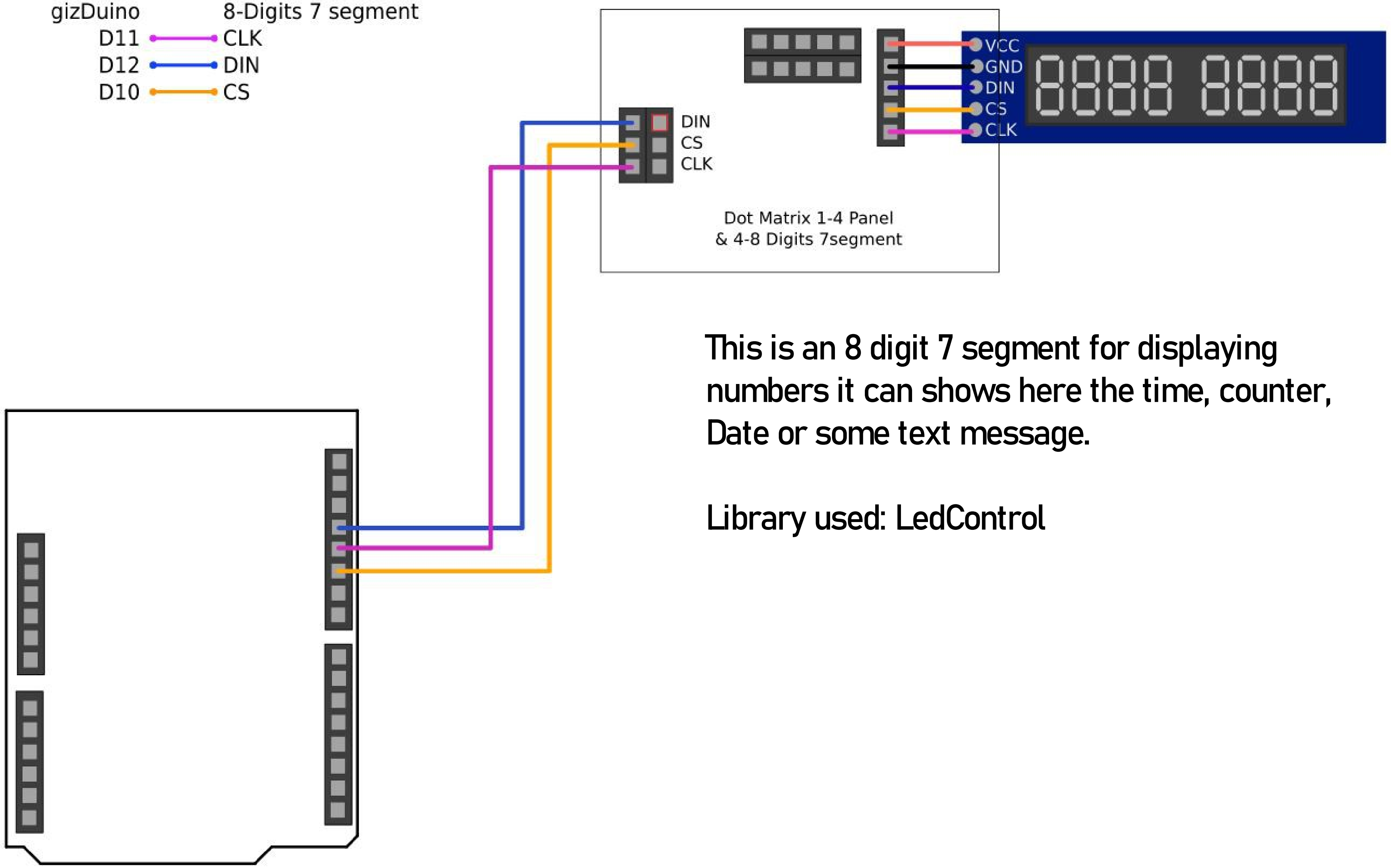
```
void loop() {
  matrix.fillScreen(LOW);
  delay(2000);
  display_message("e-gizmo Mechatronix Central");
}

void display_message(String message){
   for ( int i = 0 ; i < width * message.length() + matrix.width() - spacer; i++ ) {
    //matrix.fillScreen(LOW);
    int letter = i / width;
    int x = (matrix.width() - 1) - i % width;
    int y = (matrix.height() - 8) / 2; // center the text vertically
    while ( x + width - spacer >= 0 && letter >= 0 ) {
      if ( letter < message.length() ) {
        matrix.drawChar(x, y, message[letter], HIGH, LOW, 1); // HIGH LOW means fore
      }
      letter--;
      x -= width;
    }
    matrix.write(); // Send bitmap to display
    delay(wait/2);
  }
}
```

- In loop
- set fillscreen
- Type the display message In string

- sample of display message function.

8 digits 7 Segment Display

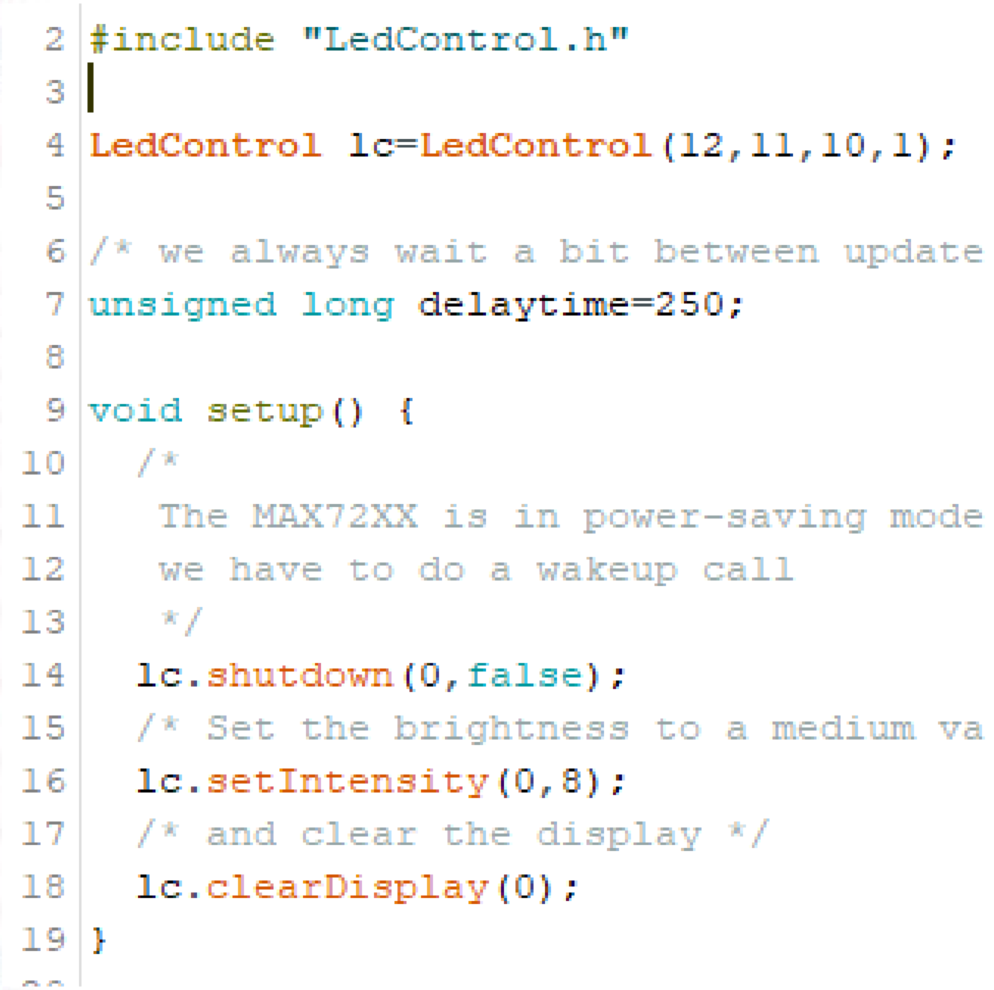
gizDuino	8-Digits 7 segment
D11	CLK
D12	DIN
D10	CS



This is an 8 digit 7 segment for displaying numbers it can shows here the time, counter, Date or some text message.

Library used: LedControl

Sample sketch for 8 digit 7 segment (setup)



```
#include "LedControl.h"

LedControl lc=LedControl(12,11,10,1);

/* we always wait a bit between update
unsigned long delaytime=250;

void setup() {
  /*
   The MAX72XX is in power-saving mode
   we have to do a wakeup call
   */
  lc.shutdown(0,false);
  /* Set the brightness to a medium va
  lc.setIntensity(0,8);
  /* and clear the display */
  lc.clearDisplay(0);
}
```

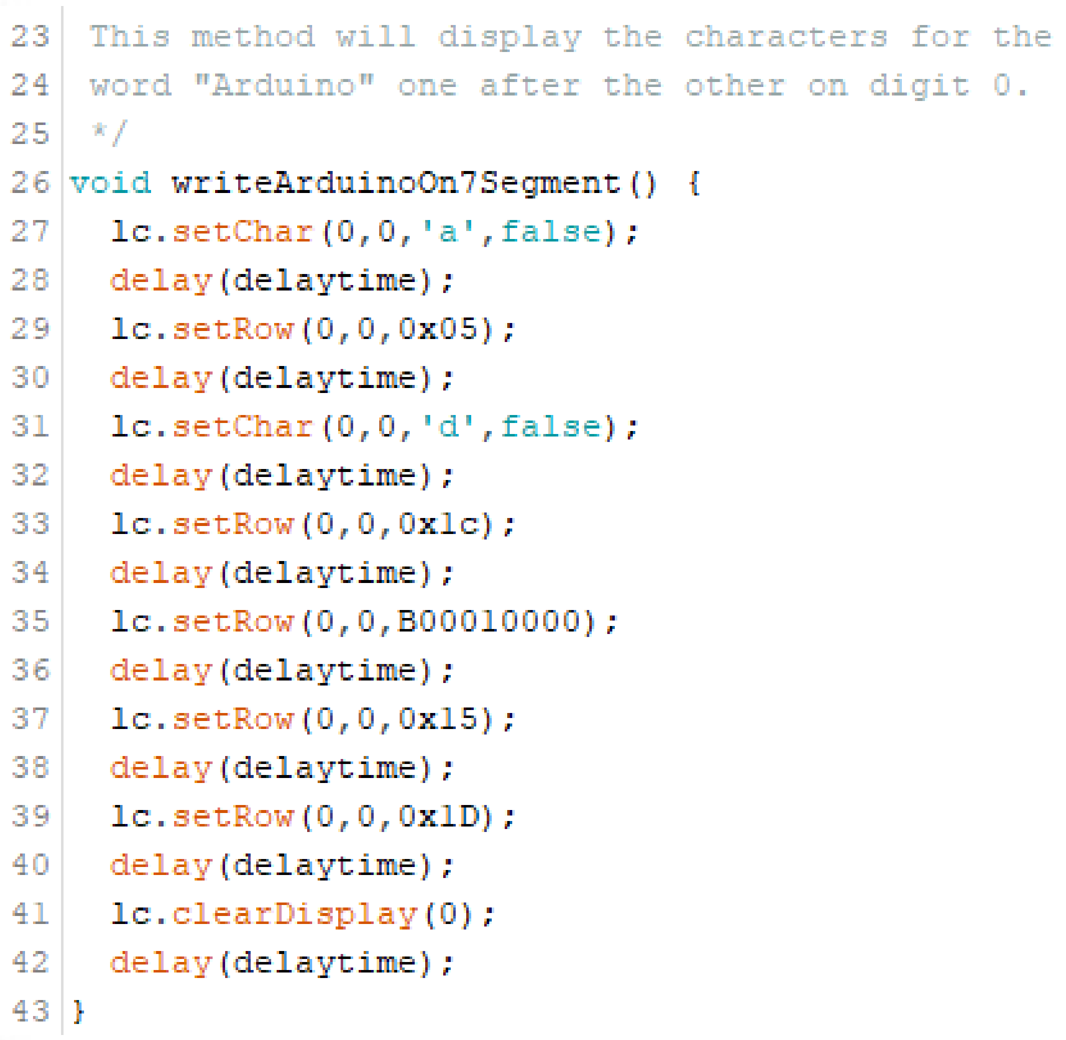
- Library used
  LedControl

- pins assignent
D11 (CLK), D12 (DIN), D10 (CS)

- delaytime

-setup
 Function shutdown, setIntensity,
ClearDisplay

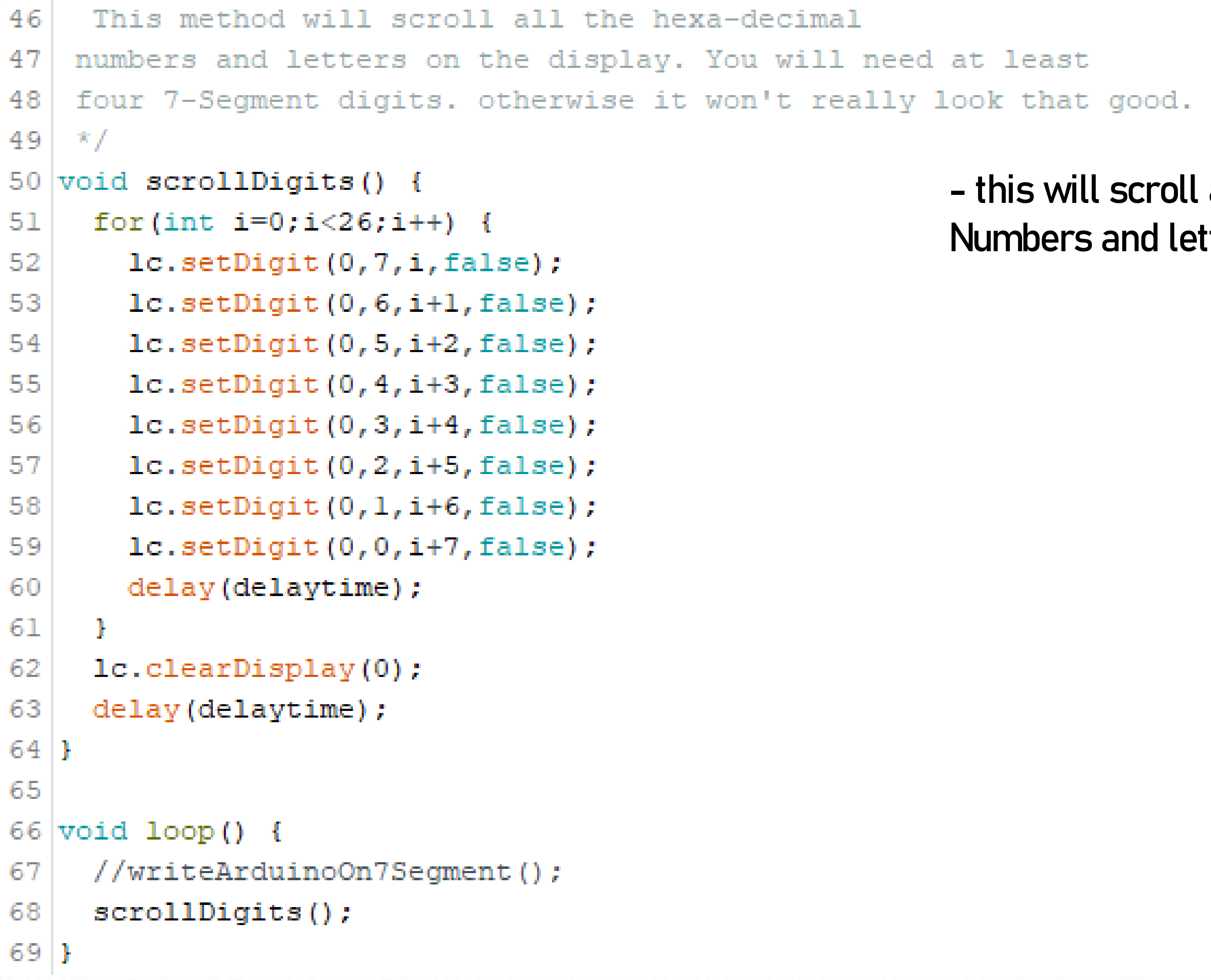
Sample sketch for 8 digit 7 segment (writeArduinoOn7sement)



```
 This method will display the characters for the
 word "Arduino" one after the other on digit 0.
 */
void writeArduinoOn7Segment() {
  lc.setChar(0,0,'a',false);
  delay(delaytime);
  lc.setRow(0,0,0x05);
  delay(delaytime);
  lc.setChar(0,0,'d',false);
  delay(delaytime);
  lc.setRow(0,0,0x1c);
  delay(delaytime);
  lc.setRow(0,0,B00010000);
  delay(delaytime);
  lc.setRow(0,0,0x15);
  delay(delaytime);
  lc.setRow(0,0,0x1D);
  delay(delaytime);
  lc.clearDisplay(0);
  delay(delaytime);
}
```

- this will display the characters For word...

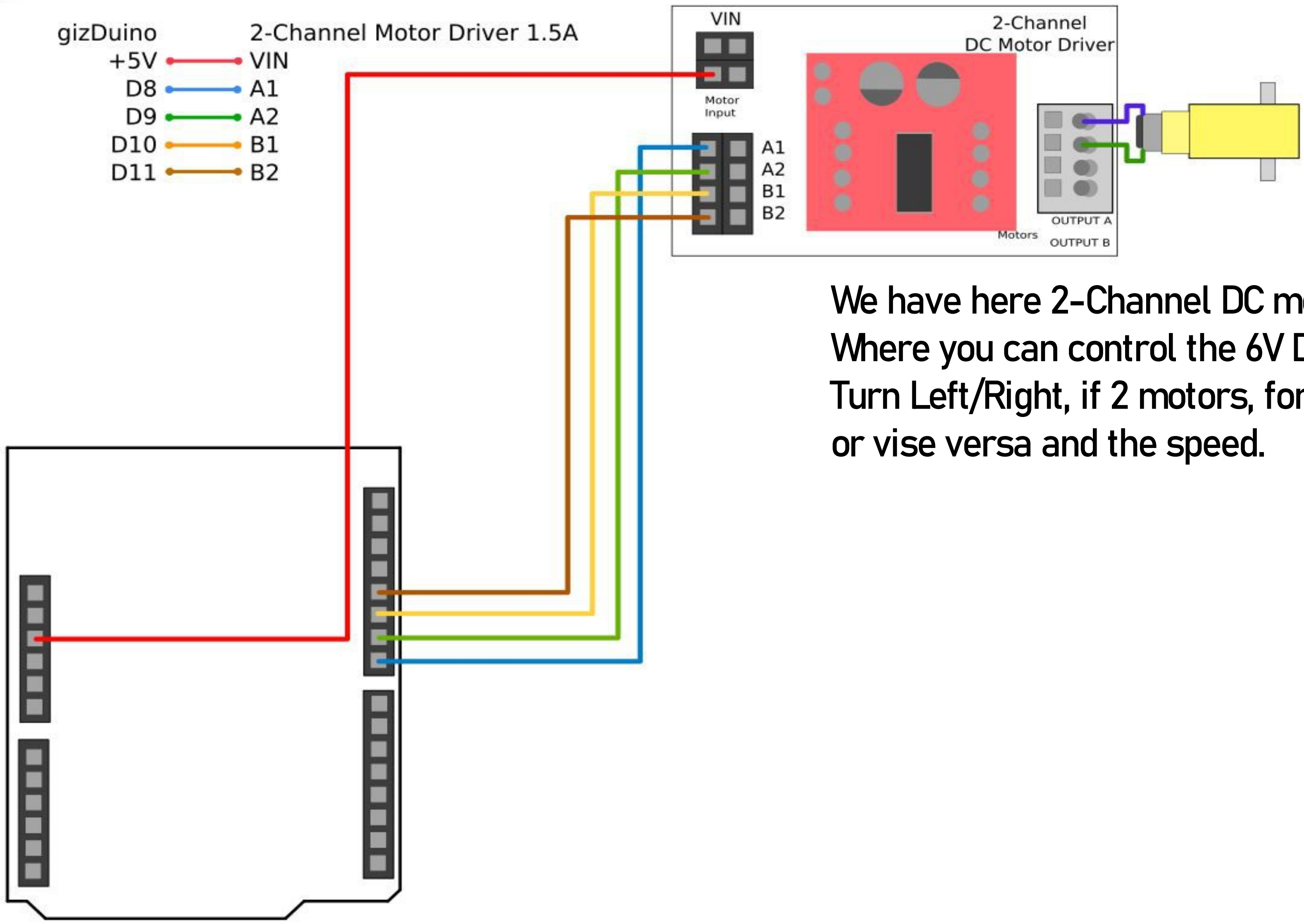
Sample sketch for 8 digit 7 segment (scrollDigits)



```
  This method will scroll all the hexa-decimal
 numbers and letters on the display. You will need at least
 four 7-Segment digits. otherwise it won't really look that good.
 */
void scrollDigits() {
  for(int i=0;i<26;i++) {
    lc.setDigit(0,7,i,false);
    lc.setDigit(0,6,i+1,false);
    lc.setDigit(0,5,i+2,false);
    lc.setDigit(0,4,i+3,false);
    lc.setDigit(0,3,i+4,false);
    lc.setDigit(0,2,i+5,false);
    lc.setDigit(0,1,i+6,false);
    lc.setDigit(0,0,i+7,false);
    delay(delaytime);
  }
  lc.clearDisplay(0);
  delay(delaytime);
}

void loop() {
  //writeArduinoOn7Segment();
  scrollDigits();
}
```

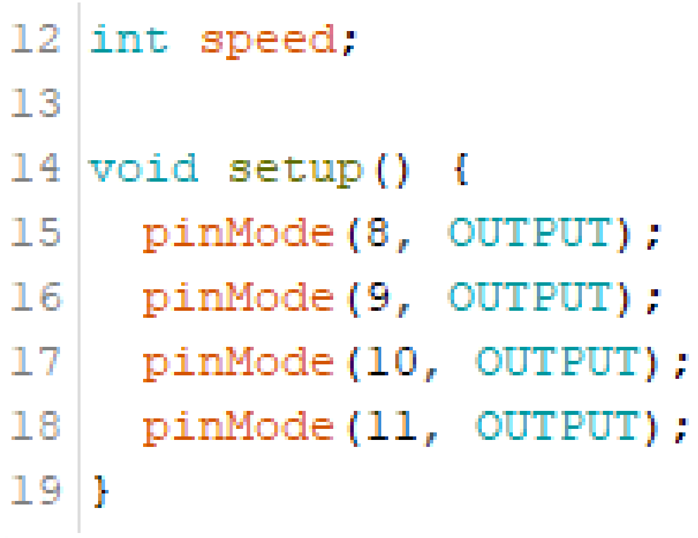
- this will scroll all the hex-decimal
Numbers and letters on the display.

2-Channel DC Motor Driver 1.5A



We have here 2-Channel DC motor Driver, Where you can control the 6V DC motors to Turn Left/Right, if 2 motors, forward/reverse or vise versa and the speed.

Sample sketch for 2-channel DC Motor Driver (setup)

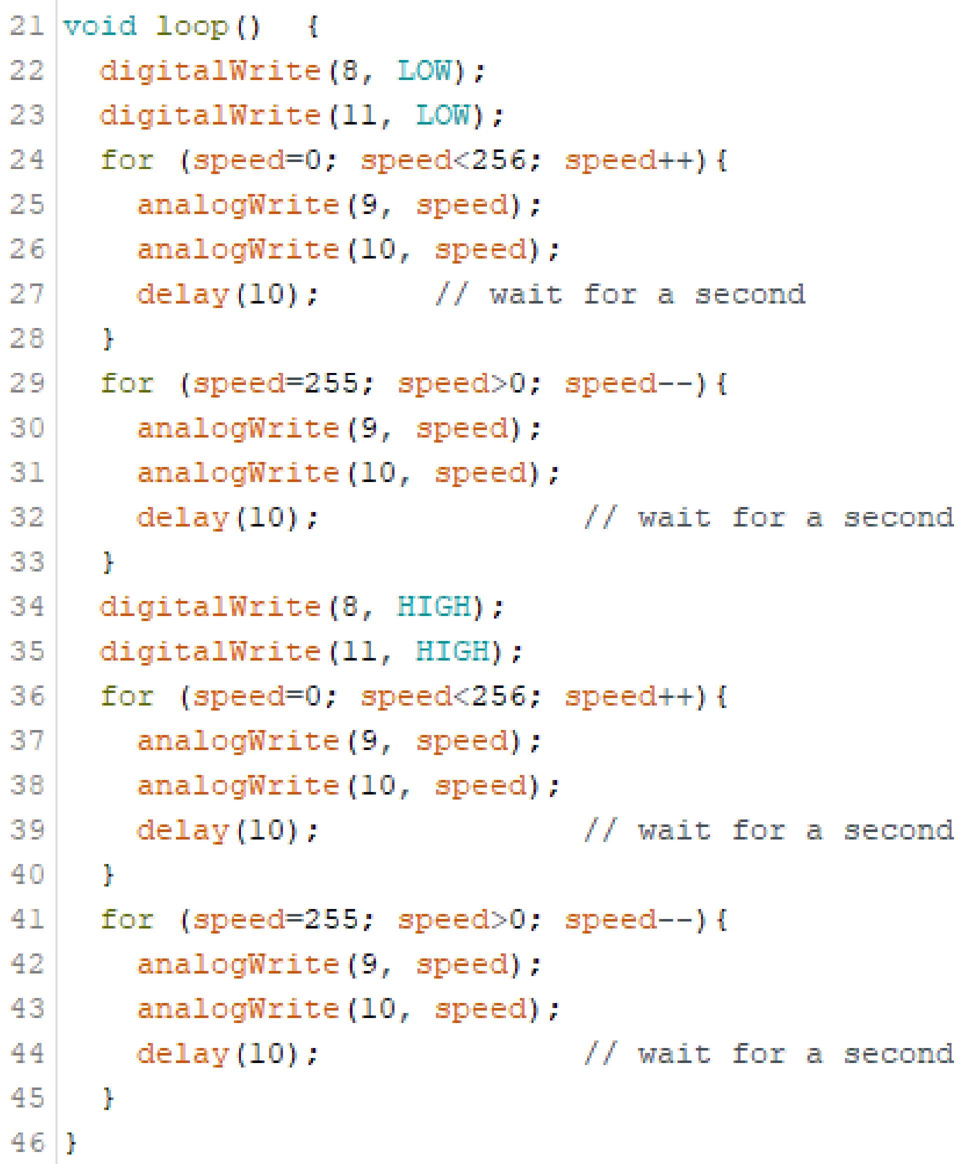


```
int speed;

void setup() {
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}
```

- variable name for speed.

- setup
Use pin 8,9,10,11 and Output mode

Sample sketch for 2-channel DC Motor Driver (loop)



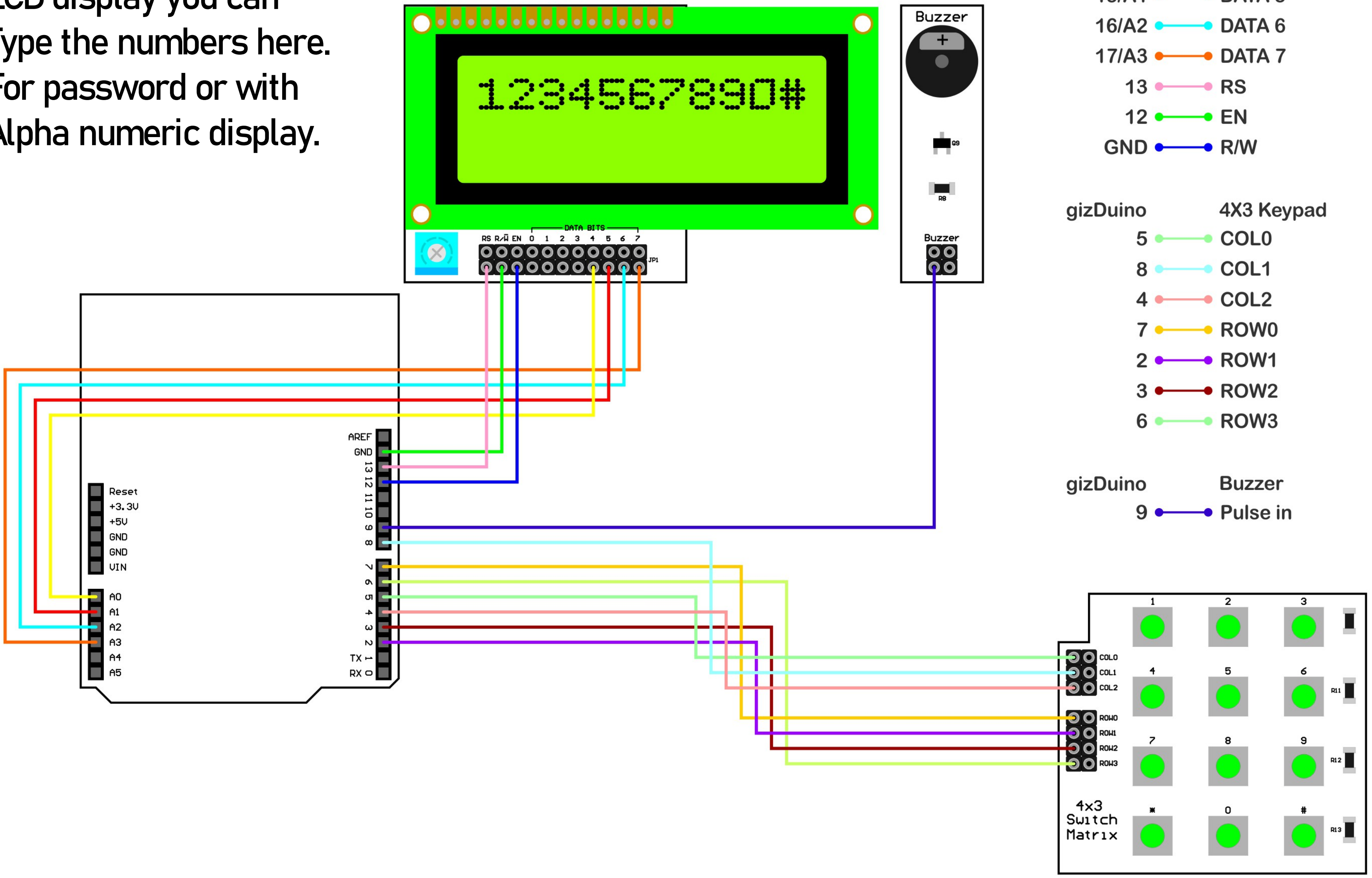
```
void loop()  {
  digitalWrite(8, LOW);
  digitalWrite(11, LOW);
  for (speed=0; speed<256; speed++){
    analogWrite(9, speed);
    analogWrite(10, speed);
    delay(10);      // wait for a second
  }
  for (speed=255; speed>0; speed--){
    analogWrite(9, speed);
    analogWrite(10, speed);
    delay(10);              // wait for a second
  }
  digitalWrite(8, HIGH);
  digitalWrite(11, HIGH);
  for (speed=0; speed<256; speed++){
    analogWrite(9, speed);
    analogWrite(10, speed);
    delay(10);              // wait for a second
  }
  for (speed=255; speed>0; speed--){
    analogWrite(9, speed);
    analogWrite(10, speed);
    delay(10);              // wait for a second
  }
}
```

- loop
  Using for loop function
  The PWM value 0 to 255
  For speed control use
  AnalogWrite.

- For changing direction
1 – HIGH or 0 – LOW
Clockwise or Counter-clockwise

4x3 Keypad Switch

By combing the buzzer and LCD display you can Type the numbers here. For password or with Alpha numeric display.

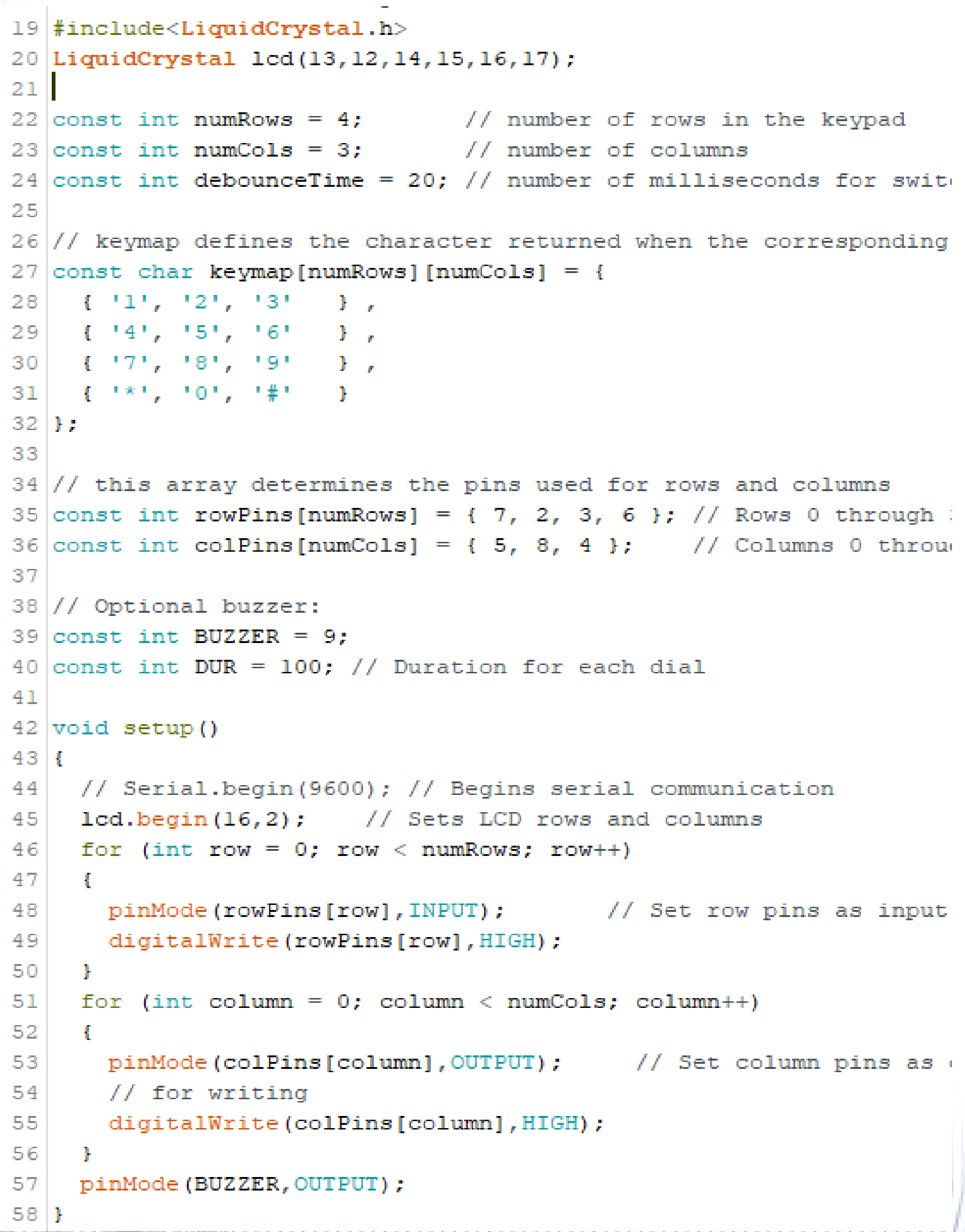


gizDuino	LCD Module
14/A0	DATA 4
15/A1	DATA 5
16/A2	DATA 6
17/A3	DATA 7
13	RS
12	EN
GND	R/W

gizDuino	4X3 Keypad
5	COL0
8	COL1
4	COL2
7	ROW0
2	ROW1
3	ROW2
6	ROW3

gizDuino	Buzzer
9	Pulse in

Sample sketch for 4x3 Keypad (setup)



```
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,14,15,16,17);

const int numRows = 4;       // number of rows in the keypad
const int numCols = 3;       // number of columns
const int debounceTime = 20; // number of milliseconds for swit

// keymap defines the character returned when the corresponding
const char keymap[numRows][numCols] = {
  { '1', '2', '3'   } ,
  { '4', '5', '6'   } ,
  { '7', '8', '9'   } ,
  { '*', '0', '#'   }
};

// this array determines the pins used for rows and columns
const int rowPins[numRows] = { 7, 2, 3, 6 }; // Rows 0 through
const int colPins[numCols] = { 5, 8, 4 };    // Columns 0 throu

// Optional buzzer:
const int BUZZER = 9;
const int DUR = 100; // Duration for each dial

void setup()
{
  // Serial.begin(9600); // Begins serial communication
  lcd.begin(16,2);    // Sets LCD rows and columns
  for (int row = 0; row < numRows; row++)
  {
    pinMode(rowPins[row],INPUT);       // Set row pins as input
    digitalWrite(rowPins[row],HIGH);
  }
  for (int column = 0; column < numCols; column++)
  {
    pinMode(colPins[column],OUTPUT);     // Set column pins as
    // for writing
    digitalWrite(colPins[column],HIGH);
  }
  pinMode(BUZZER,OUTPUT);
}
```

- Library used
  LiquidCrystal

- assigned pins 13,12,14,15,16,17
  (see the wiring diagram)

- number of Rows and Columns

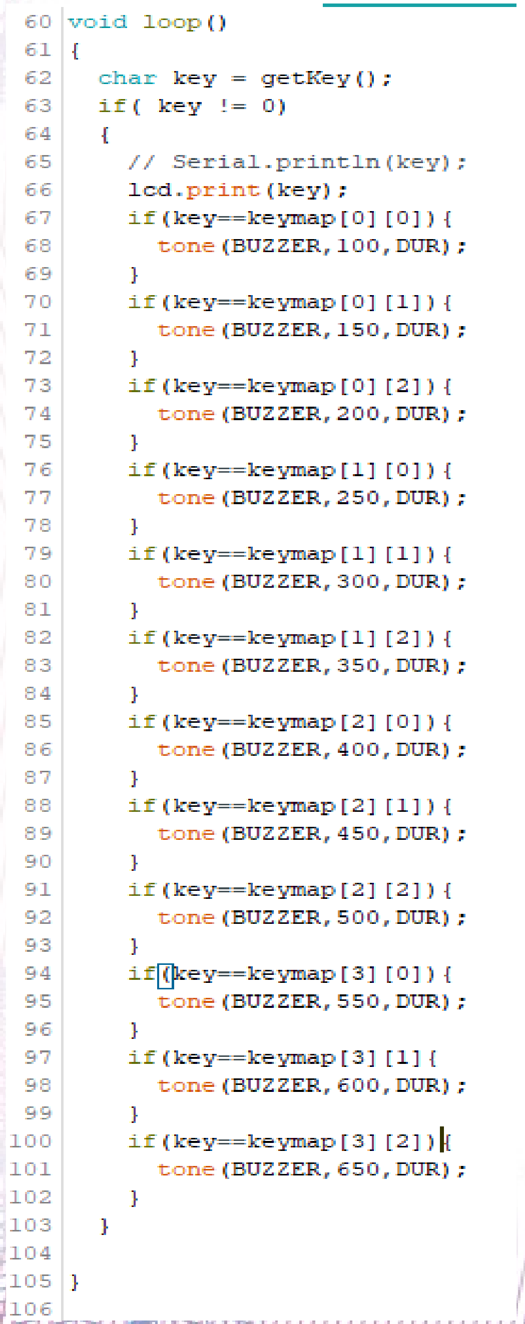
- key mapping

- buzzer pin

- set begin(16,2) for 16x2 lcd
  For 20x4 lcd – set begin(20,4)

- set all the row pin to input while column Pin to output. And all high-state (Normally High)

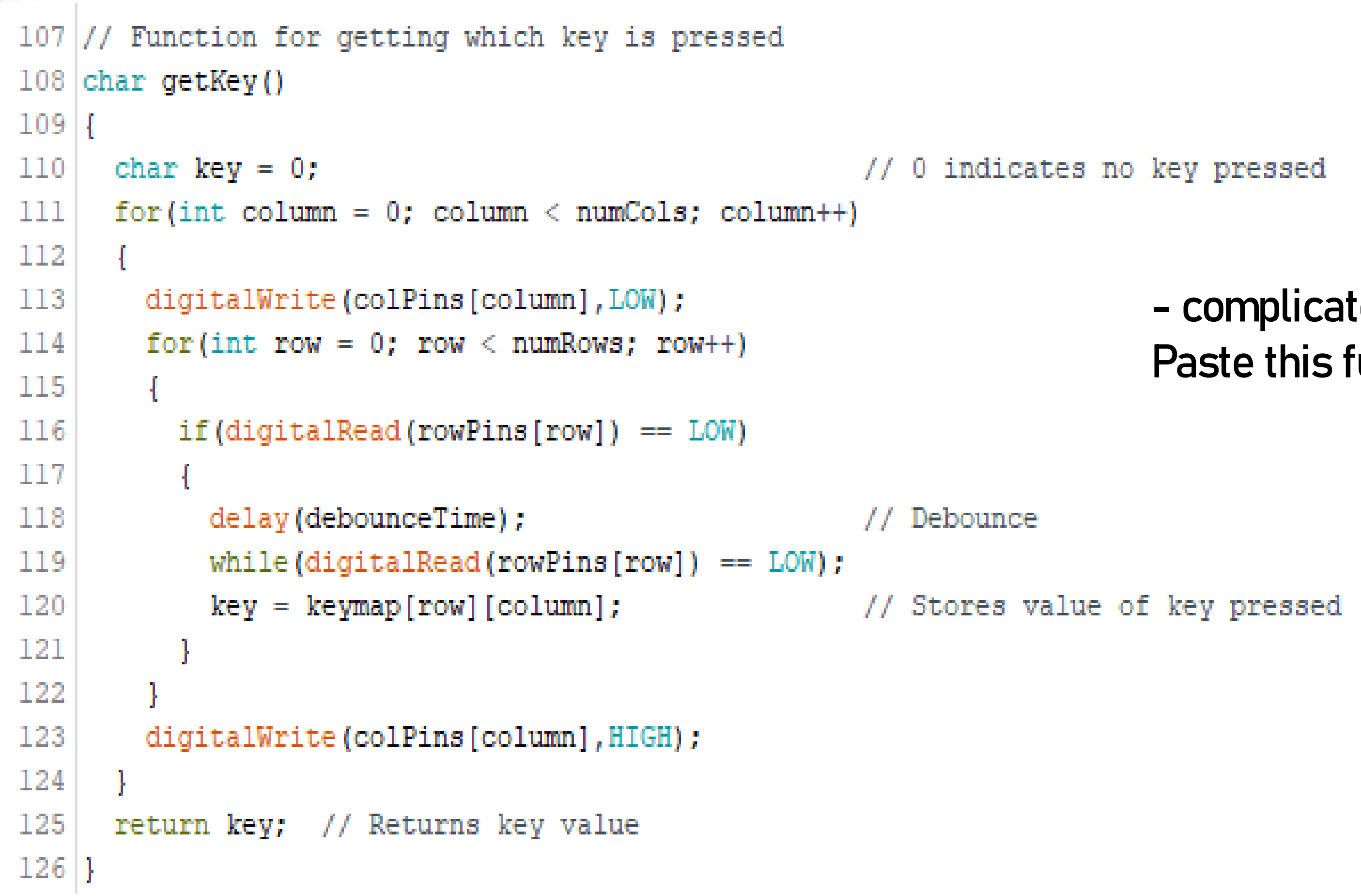
Sample sketch for 4x3 Keypad (loop)



```
void loop()
{
  char key = getKey();
  if( key != 0)
  {
    // Serial.println(key);
    lcd.print(key);
    if(key==keymap[0][0]){
      tone(BUZZER,100,DUR);
    }
    if(key==keymap[0][1]){
      tone(BUZZER,150,DUR);
    }
    if(key==keymap[0][2]){
      tone(BUZZER,200,DUR);
    }
    if(key==keymap[1][0]){
      tone(BUZZER,250,DUR);
    }
    if(key==keymap[1][1]){
      tone(BUZZER,300,DUR);
    }
    if(key==keymap[1][2]){
      tone(BUZZER,350,DUR);
    }
    if(key==keymap[2][0]){
      tone(BUZZER,400,DUR);
    }
    if(key==keymap[2][1]){
      tone(BUZZER,450,DUR);
    }
    if(key==keymap[2][2]){
      tone(BUZZER,500,DUR);
    }
    if(key==keymap[3][0]){
      tone(BUZZER,550,DUR);
    }
    if(key==keymap[3][1]{
      tone(BUZZER,600,DUR);
    }
    if(key==keymap[3][2]){
      tone(BUZZER,650,DUR);
    }
  }

}
```

- loop
= using if condition to get the pressed key (numbers/ symbol)

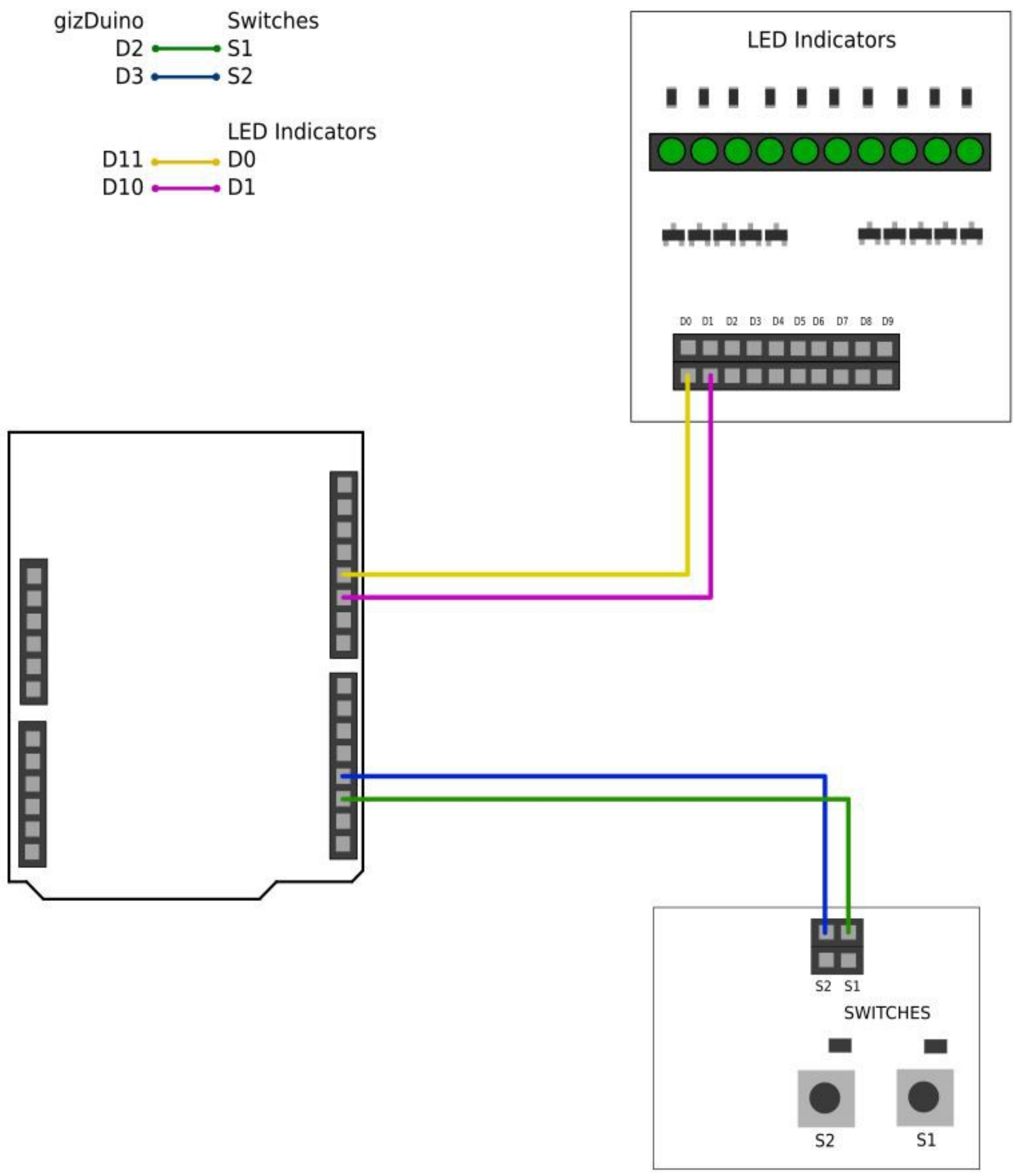
Sample sketch for 4x3 Keypad (getKey functions)



```
// Function for getting which key is pressed
char getKey()
{
  char key = 0;                                  // 0 indicates no key pressed
  for(int column = 0; column < numCols; column++)
  {
    digitalWrite(colPins[column],LOW);
    for(int row = 0; row < numRows; row++)
    {
      if(digitalRead(rowPins[row]) == LOW)
      {
        delay(debounceTime);                     // Debounce
        while(digitalRead(rowPins[row]) == LOW);
        key = keymap[row][column];               // Stores value of key pressed
      }
    }
    digitalWrite(colPins[column],HIGH);
  }
  return key;  // Returns key value
}
```

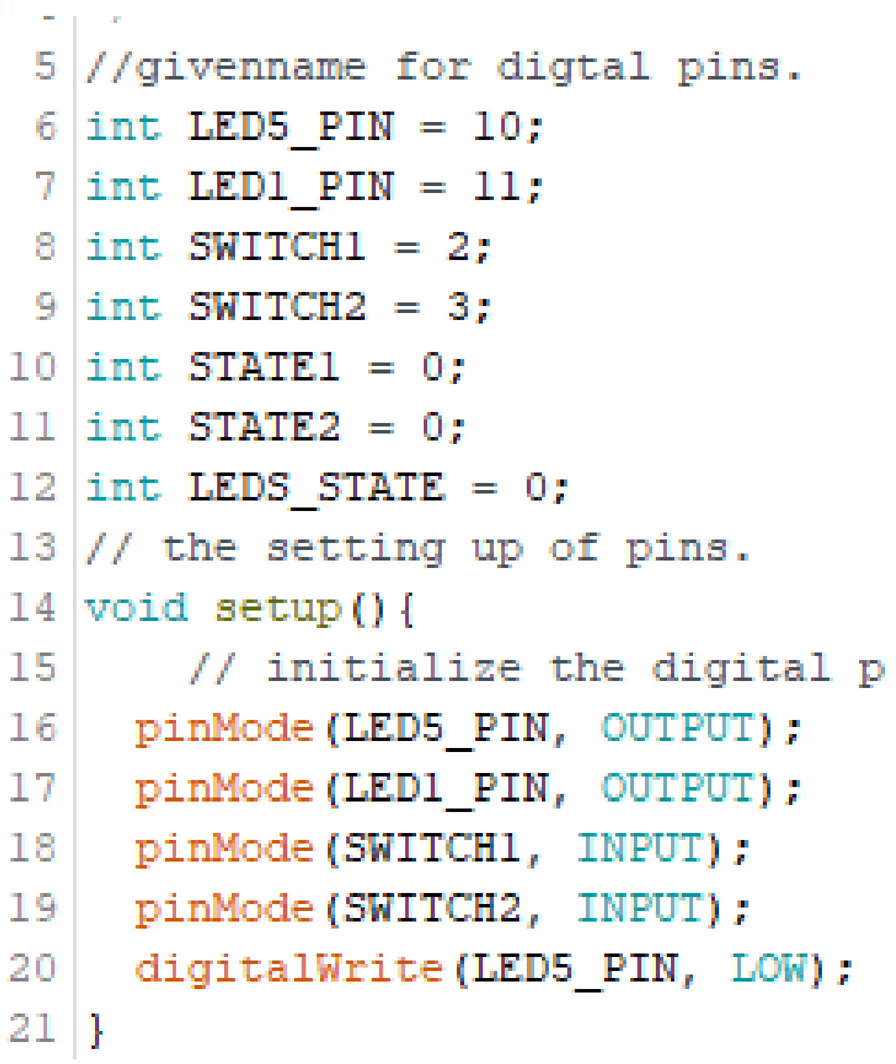
- complicated but you can copy and Paste this function to use.

Switches (2 Push Button)



Understanding how to use button as
A real switch. For turning LED light
On/off. If we say push button, every
Pressed the LED turns ON and if it is
Release the LED tuns OFF.
While latching is when you press
Once the button the LED state
Remain on HIGH, that's why if you
Press again the LED state is LOW.

Sample sketch for Latching & push button (setup)

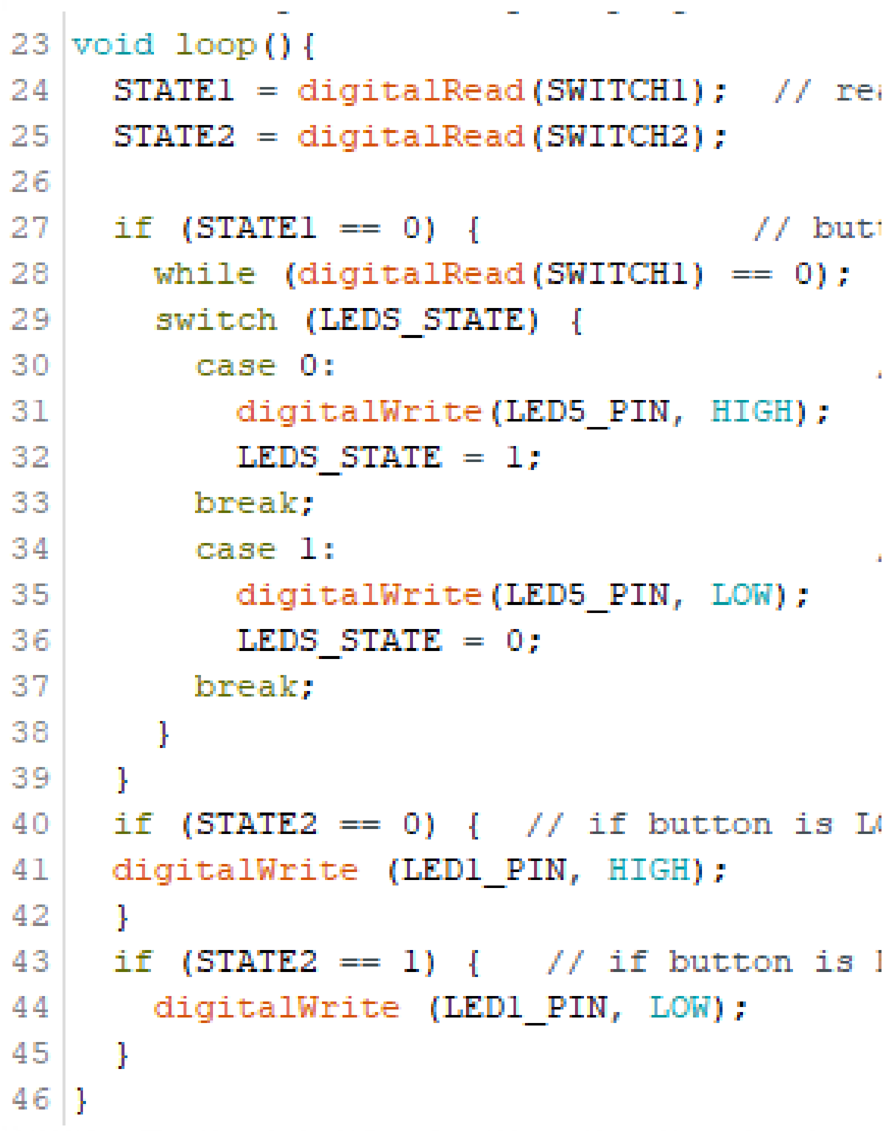


```
//givenname for digtal pins.
int LED5_PIN = 10;
int LED1_PIN = 11;
int SWITCH1 = 2;
int SWITCH2 = 3;
int STATE1 = 0;
int STATE2 = 0;
int LEDS_STATE = 0;
// the setting up of pins.
void setup(){
    // initialize the digital p
  pinMode(LED5_PIN, OUTPUT);
  pinMode(LED1_PIN, OUTPUT);
  pinMode(SWITCH1, INPUT);
  pinMode(SWITCH2, INPUT);
  digitalWrite(LED5_PIN, LOW);
}
```

- assigned pins for button and LEDs
  (see the wiring diagram)

- setup the pins

Sample sketch for Latching & push button (loop)



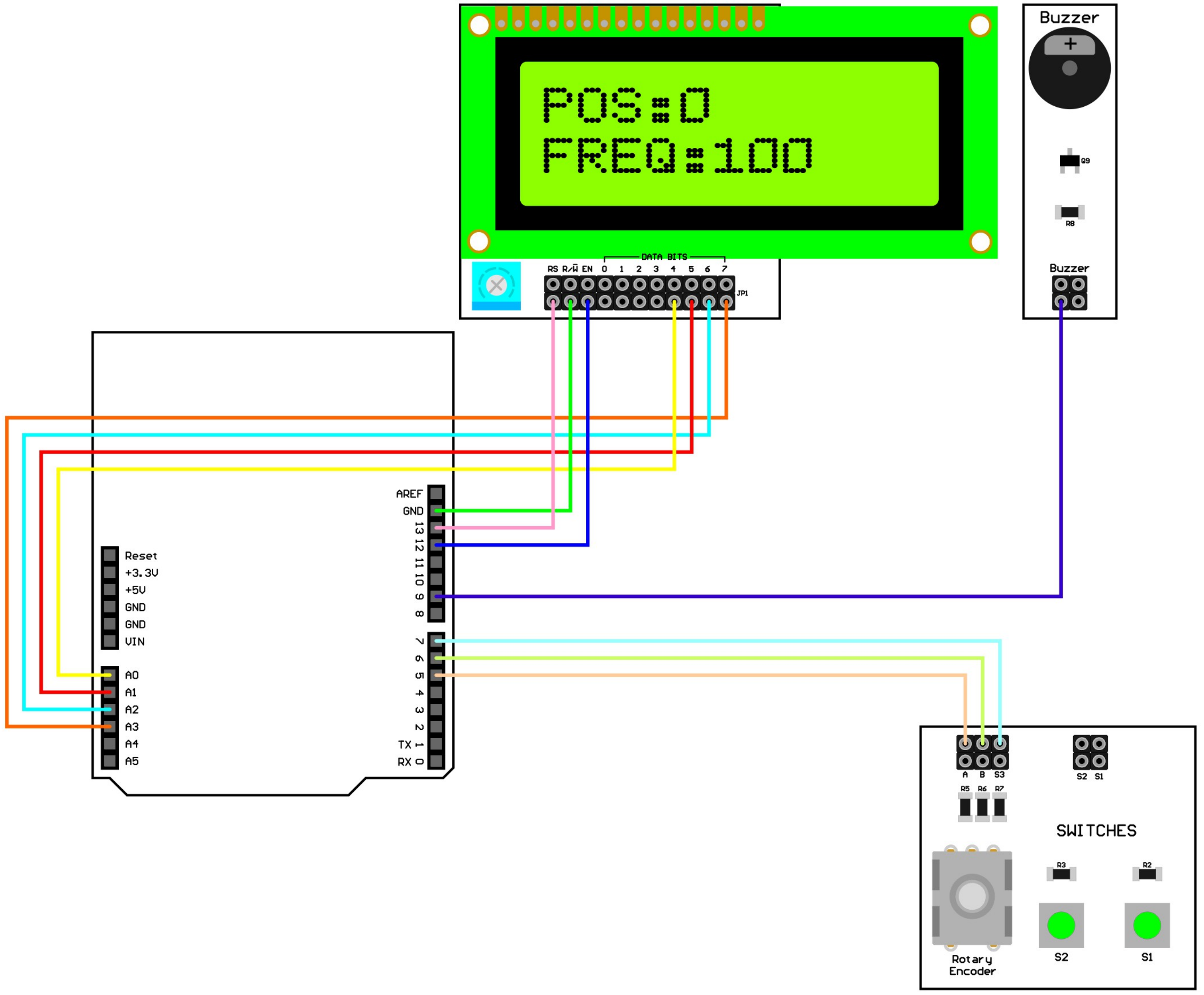
```
void loop(){
  STATE1 = digitalRead(SWITCH1);   // re
  STATE2 = digitalRead(SWITCH2);

  if (STATE1 == 0) {                // but
    while (digitalRead(SWITCH1) == 0);
    switch (LEDS_STATE) {
      case 0:
        digitalWrite(LED5_PIN, HIGH);
        LEDS_STATE = 1;
      break;
      case 1:
        digitalWrite(LED5_PIN, LOW);
        LEDS_STATE = 0;
      break;
    }
  }
  if (STATE2 == 0) {   // if button is L
  digitalWrite (LED1_PIN, HIGH);
  }
  if (STATE2 == 1) {    // if button is I
    digitalWrite (LED1_PIN, LOW);
  }
}
```

- loop
  If you are using buttons/switch
  Use digitalRead to get the data.

- if condition to read if there's a changed.
  Switch case for Latching
  And if-if condition for Push buttons.
  Or if-if else.

Note: Do not use if-else condition here.
Your program will not work properly.

Rotary Encoder



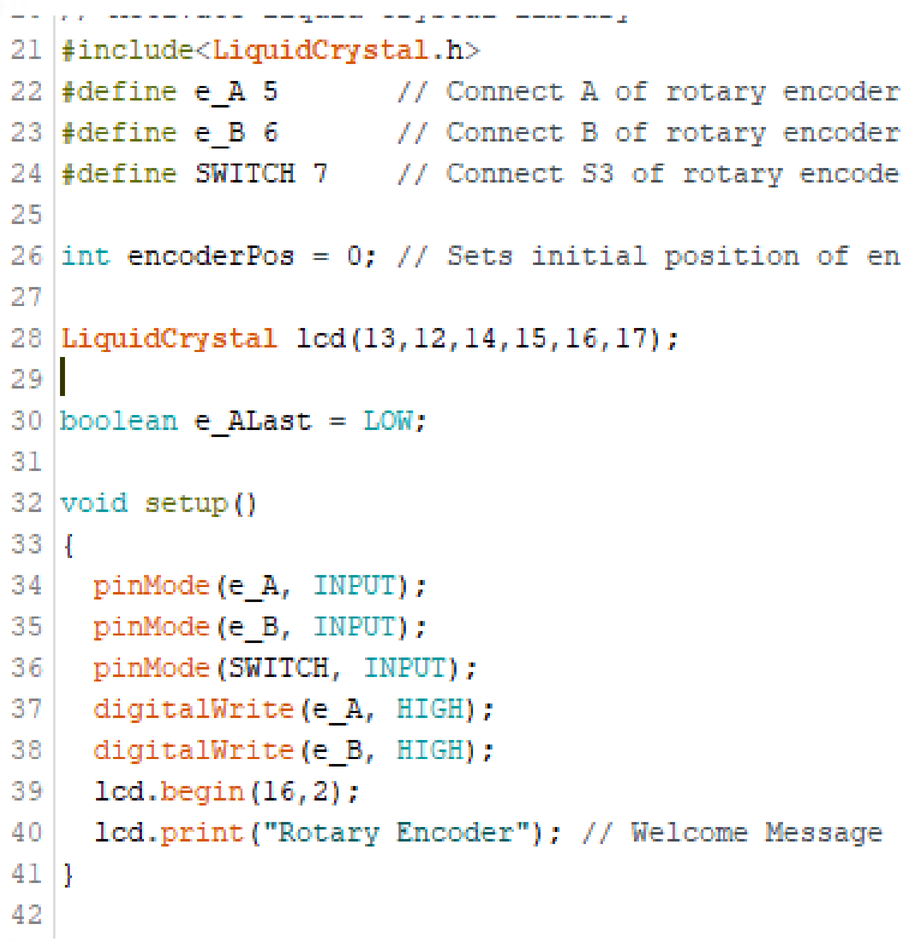
gizDuino	LCD Module
14/A0	DATA 4
15/A1	DATA 5
16/A2	DATA 6
17/A3	DATA 7
13	RS
12	EN
GND	R/W

gizDuino	Rotary Encoder
5	A
6	B
7	S3

gizDuino	Buzzer
9	Pulse in

Rotary Encoder
Is good for switching
,counter, speed control,
Volume controls,
Etc.

Sample sketch for Rotary Encoder (setup)



```
#include<LiquidCrystal.h>
#define e_A 5        // Connect A of rotary encoder
#define e_B 6        // Connect B of rotary encoder
#define SWITCH 7     // Connect S3 of rotary encode

int encoderPos = 0; // Sets initial position of en

LiquidCrystal lcd(13,12,14,15,16,17);

boolean e_ALast = LOW;

void setup()
{
  pinMode(e_A, INPUT);
  pinMode(e_B, INPUT);
  pinMode(SWITCH, INPUT);
  digitalWrite(e_A, HIGH);
  digitalWrite(e_B, HIGH);
  lcd.begin(16,2);
  lcd.print("Rotary Encoder"); // Welcome Message
}
```

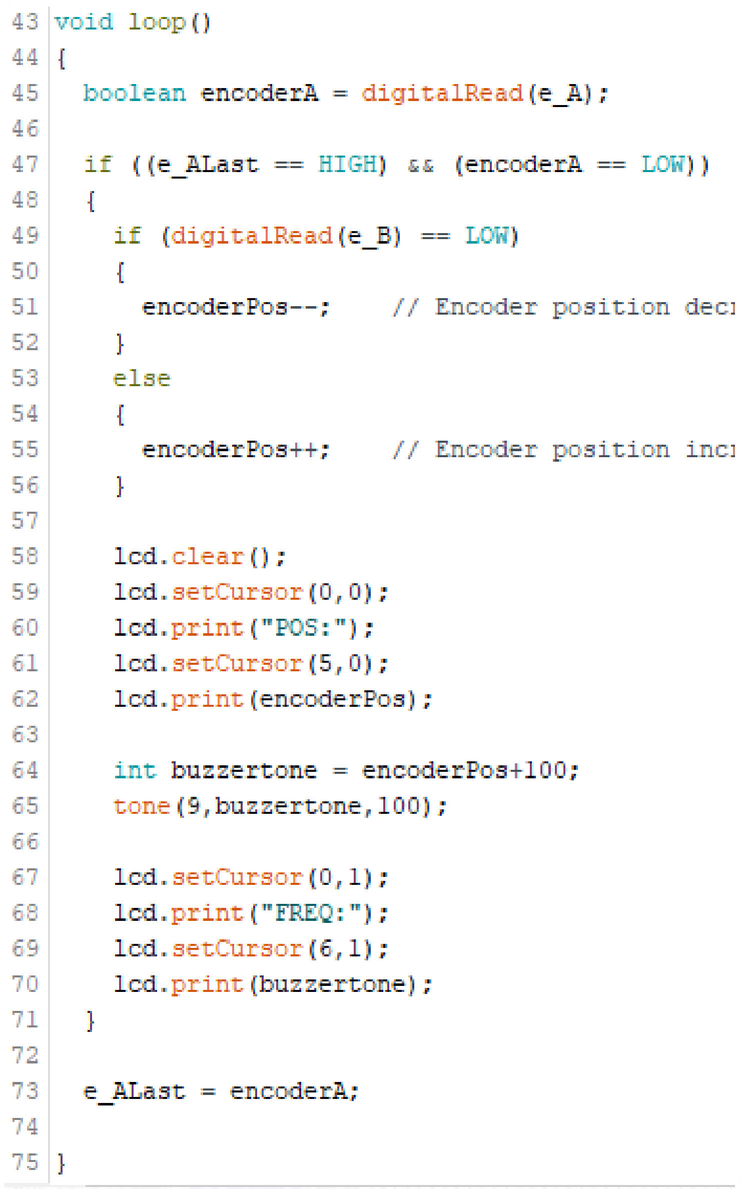
- Library used
  LiquidCrystal
- define pins 5, 6,7 (see the wiring)

- initial position 0
-lcd pin assigment (see the wiring)

- setup
  Pins set to input and on high-state.

- lcd begin (16,2) for lcd

Sample sketch for Rotary Encoder (loop)



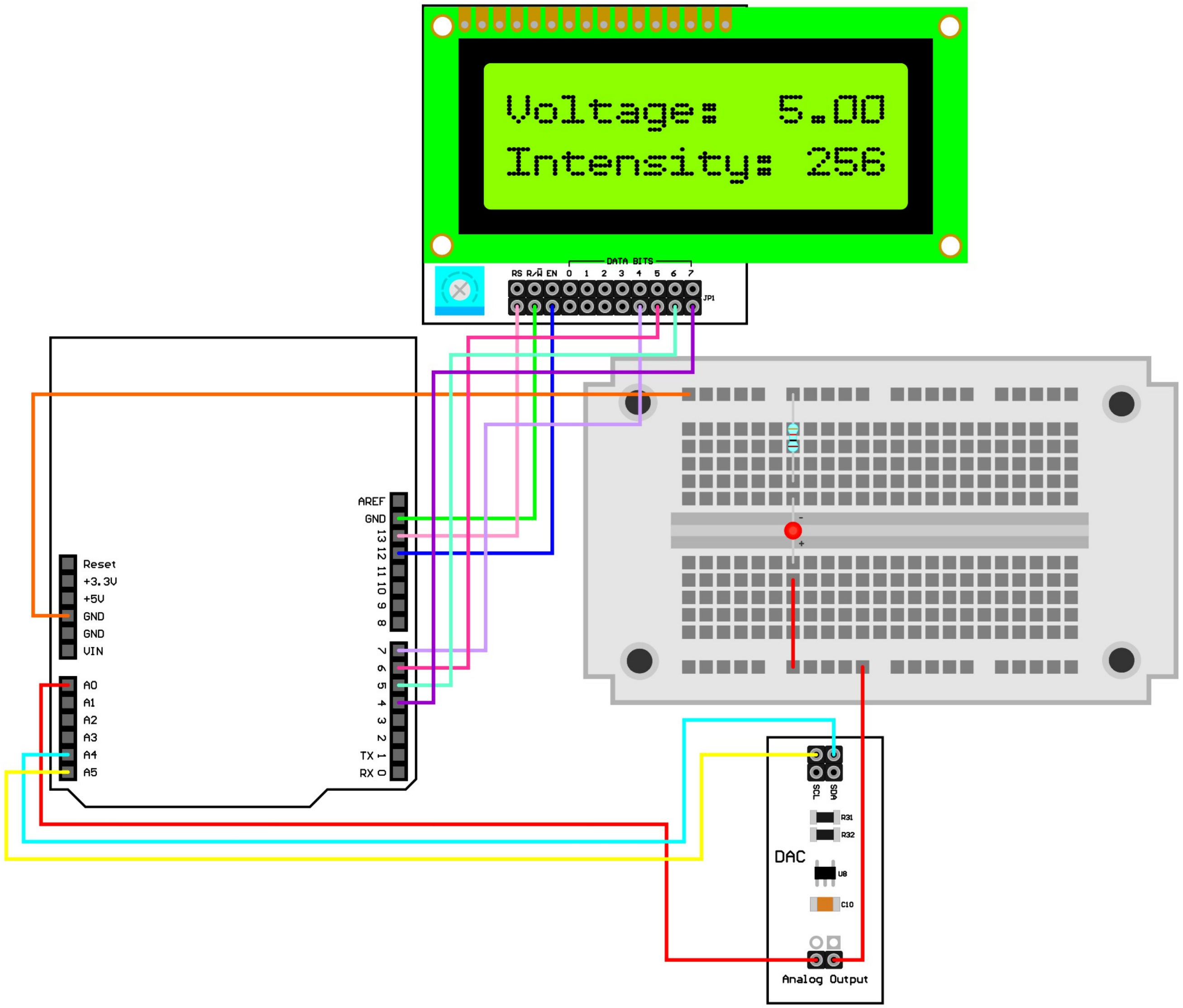
```
void loop()
{
  boolean encoderA = digitalRead(e_A);

  if ((e_ALast == HIGH) && (encoderA == LOW))
  {
    if (digitalRead(e_B) == LOW)
    {
      encoderPos--;     // Encoder position dec
    }
    else
    {
      encoderPos++;     // Encoder position inc
    }

    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("POS:");
    lcd.setCursor(5,0);
    lcd.print(encoderPos);

    int buzzertone = encoderPos+100;
    tone(9,buzzertone,100);

    lcd.setCursor(0,1);
    lcd.print("FREQ:");
    lcd.setCursor(6,1);
    lcd.print(buzzertone);
  }

  e_ALast = encoderA;

}
```

- loop
  encoder code using if condition
  For decreasing and increasing value.

- and the value displayed on the LCD screen.

Digital-to-Analog Converter or DAC

Convert Digital to Analog.
Voltage or intensity application



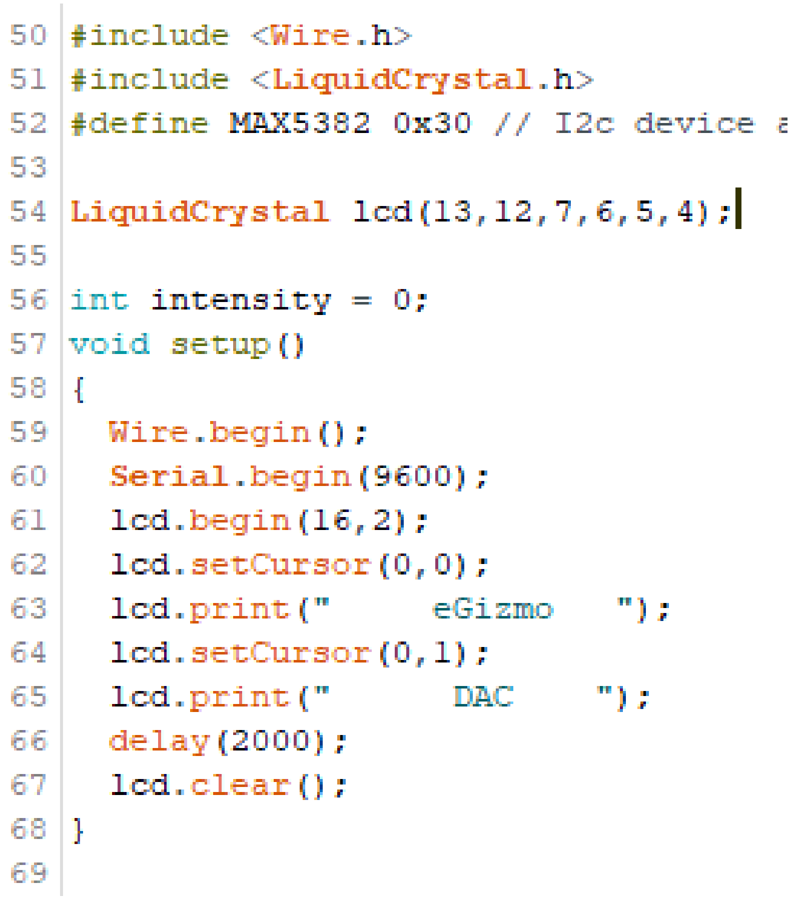
gizDuino	LCD Module
7	DATA 4
6	DATA 5
5	DATA 6
4	DATA 7
13	RS
12	EN
GND	R/W

gizDuino	DAC
A0	Analog Output
A4	SDA
A5	SCL

Items Used:

- 1k 1/4 watt Resistor
- 5mm Red LED
- male to male connecting wires

Sample sketch for DAC (setup)



```
#include <Wire.h>
#include <LiquidCrystal.h>
#define MAX5382 0x30 // I2c device a

LiquidCrystal lcd(13,12,7,6,5,4);

int intensity = 0;
void setup()
{
  Wire.begin();
  Serial.begin(9600);
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("     eGizmo   ");
  lcd.setCursor(0,1);
  lcd.print("      DAC    ");
  delay(2000);
  lcd.clear();
}
```

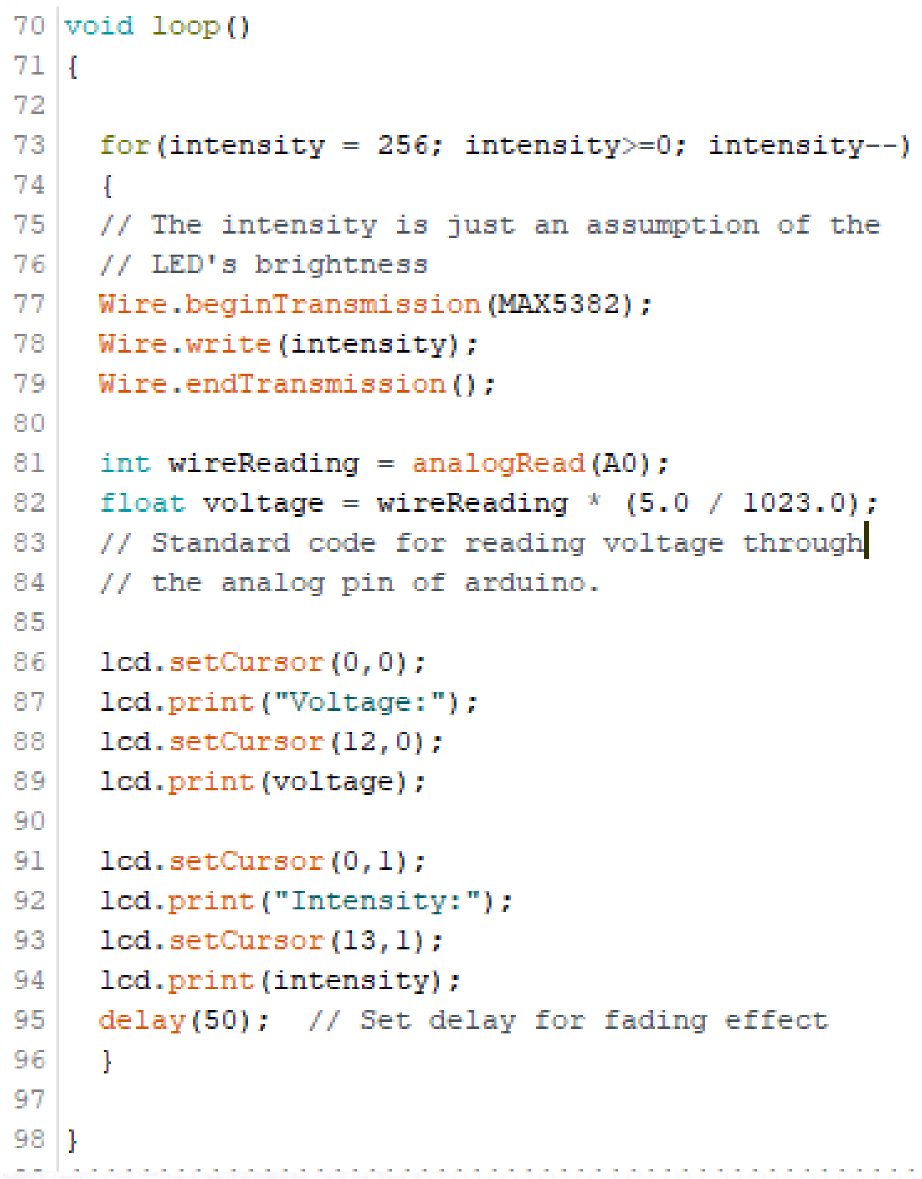
- Library used
  Wire, LiquidCrystal

- define i2x address of the device

- set the intensity to 0

- setup

Sample sketch for DAC (loop)



```
void loop()
{

  for(intensity = 256; intensity>=0; intensity--)
  {
  // The intensity is just an assumption of the
  // LED's brightness
  Wire.beginTransmission(MAX5382);
  Wire.write(intensity);
  Wire.endTransmission();

  int wireReading = analogRead(A0);
  float voltage = wireReading * (5.0 / 1023.0);
  // Standard code for reading voltage through
  // the analog pin of arduino.

  lcd.setCursor(0,0);
  lcd.print("Voltage:");
  lcd.setCursor(12,0);
  lcd.print(voltage);

  lcd.setCursor(0,1);
  lcd.print("Intensity:");
  lcd.setCursor(13,1);
  lcd.print(intensity);
  delay(50);  // Set delay for fading effect
  }

}
```

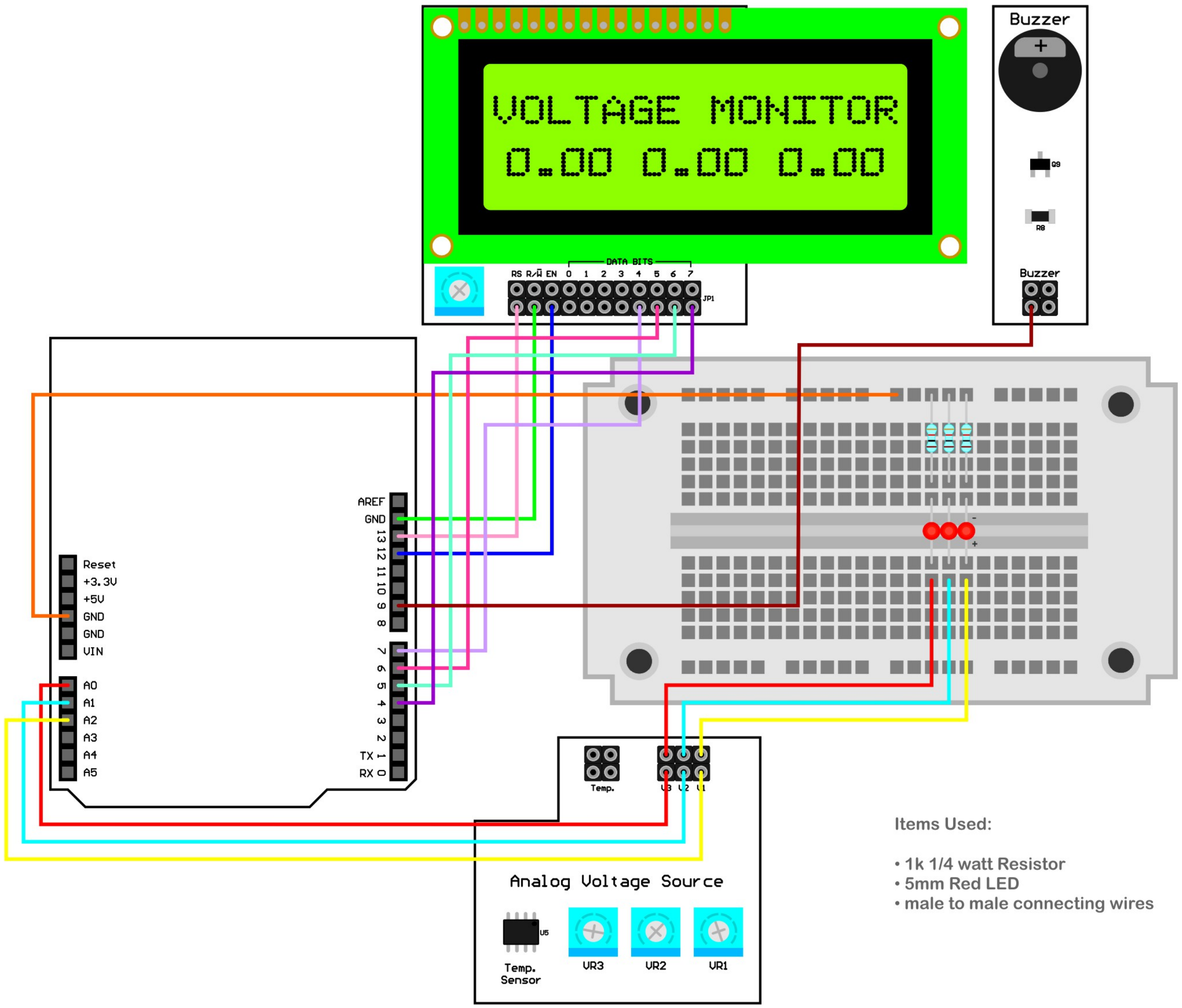
- loop

- reading the analog 0

- Converting into voltage

- lcd display

Analog Voltage Source



gizDuino	LCD Module
7	DATA 4
6	DATA 5
5	DATA 6
4	DATA 7
13	RS
12	EN
GND	R/W

gizDuino	A.V.S
A2	V1
A1	V2
A0	V3

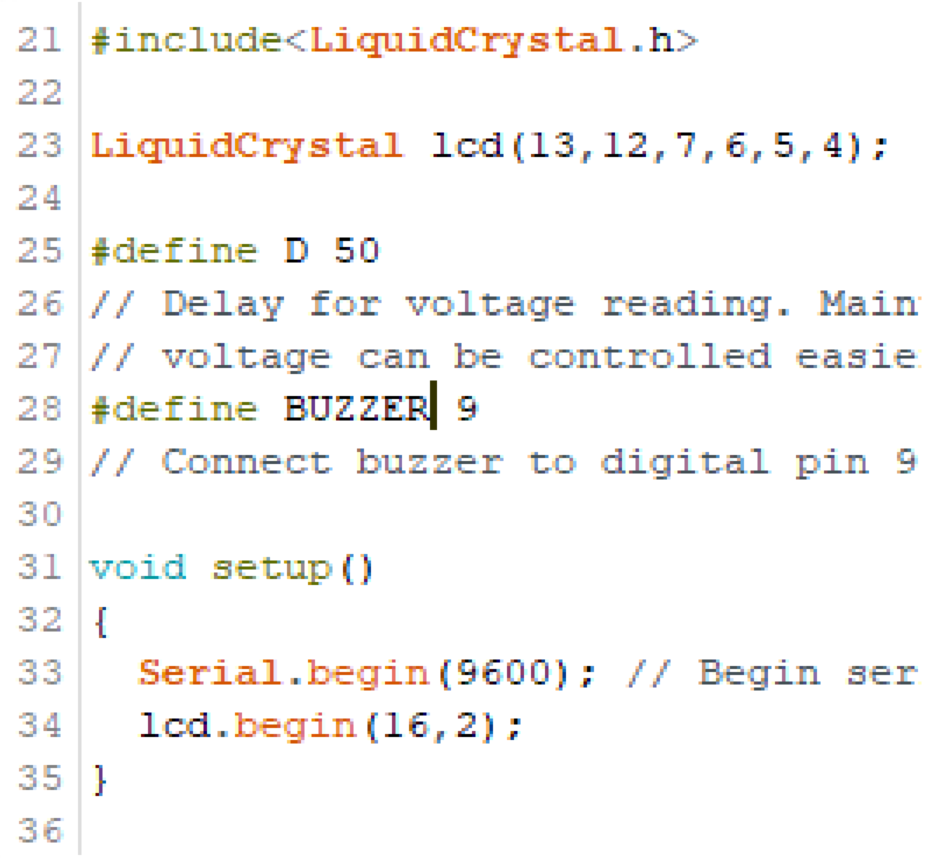
gizDuino	Buzzer
9	Pulse in

Items Used:

- 1k 1/4 watt Resistor
- 5mm Red LED
- male to male connecting wires

We use the trimmer/ Potentiometer as a voltage Adjustment, LED light Intensity control, volume, Analog reading etc.

Sample sketch for Anaglog Voltage Source (setup)

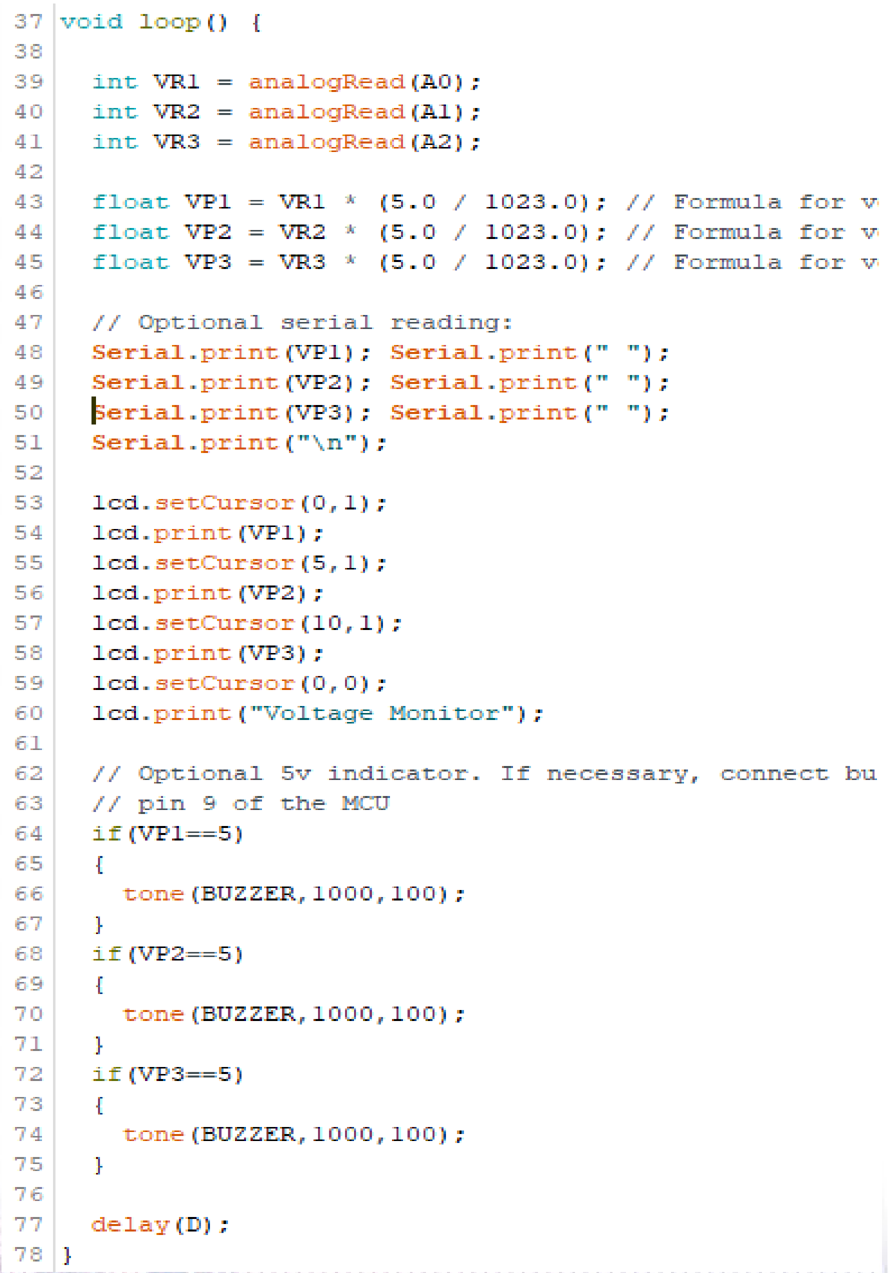


```
#include<LiquidCrystal.h>

LiquidCrystal lcd(13,12,7,6,5,4);

#define D 50
// Delay for voltage reading. Main
// voltage can be controlled easie
#define BUZZER 9
// Connect buzzer to digital pin 9

void setup()
{
  Serial.begin(9600); // Begin ser
  lcd.begin(16,2);
}

```

-Library used
 LiquidCrystal
- pin assignments (see the wiring)
- D- delay
- buzzer pin

-setup
Baudrate 9600
Lcd set to 16,2

Sample sketch for Anaglog Voltage Source (loop)



```
void loop() {

  int VR1 = analogRead(A0);
  int VR2 = analogRead(A1);
  int VR3 = analogRead(A2);

  float VP1 = VR1 * (5.0 / 1023.0); // Formula for v
  float VP2 = VR2 * (5.0 / 1023.0); // Formula for v
  float VP3 = VR3 * (5.0 / 1023.0); // Formula for v

  // Optional serial reading:
  Serial.print(VP1); Serial.print(" ");
  Serial.print(VP2); Serial.print(" ");
  Serial.print(VP3); Serial.print(" ");
  Serial.print("\n");

  lcd.setCursor(0,1);
  lcd.print(VP1);
  lcd.setCursor(5,1);
  lcd.print(VP2);
  lcd.setCursor(10,1);
  lcd.print(VP3);
  lcd.setCursor(0,0);
  lcd.print("Voltage Monitor");

  // Optional 5v indicator. If necessary, connect bu
  // pin 9 of the MCU
  if(VP1==5)
  {
    tone(BUZZER,1000,100);
  }
  if(VP2==5)
  {
    tone(BUZZER,1000,100);
  }
  if(VP3==5)
  {
    tone(BUZZER,1000,100);
  }

  delay(D);
}
```

- loop
- used analogRead to get the analog Data in analog pins

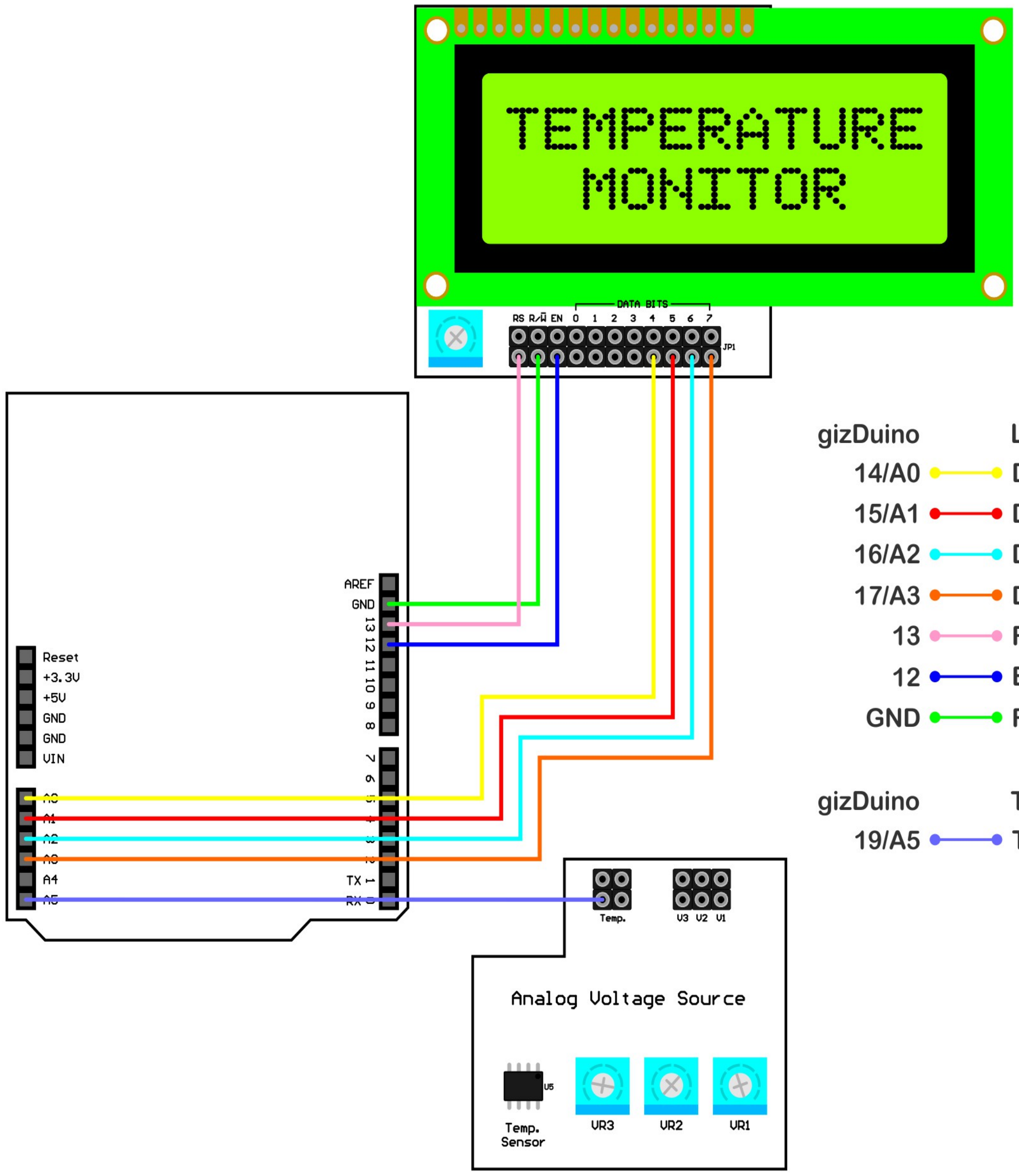
- Formula/convert into voltage
- print the value

- lcd display

- if condition

If the value exceed to 5V the buzzer will Sound.

Temperature sensor LM34

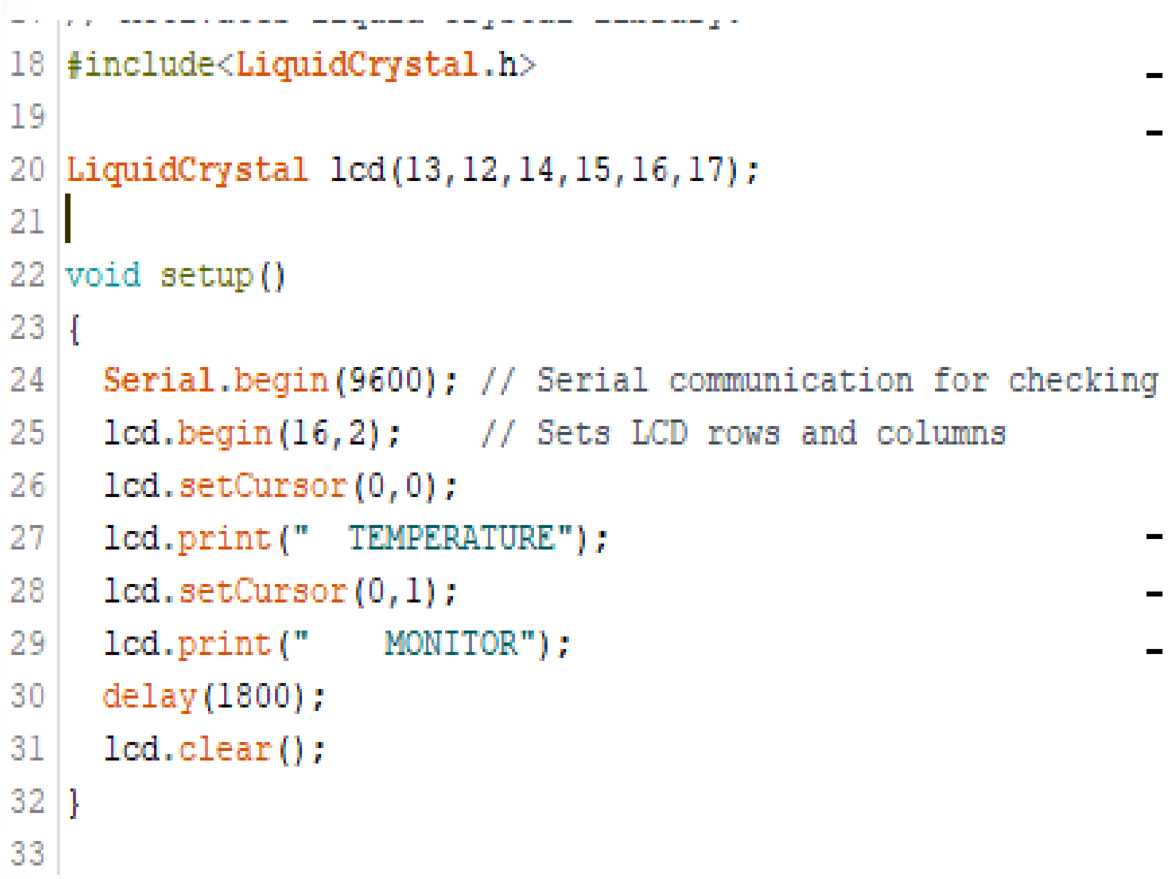


gizDuino	LCD Module
14/A0	DATA 4
15/A1	DATA 5
16/A2	DATA 6
17/A3	DATA 7
13	RS
12	EN
GND	R/W

gizDuino	Temp. Sensor
19/A5	Temperature Out

We have here an analog temperature Sensor LM34 (more on Farenheight Calibrated) display, we can also convert it to Kelvin, or Celcius formonitoring.

Sample sketch for LM34 (setup)

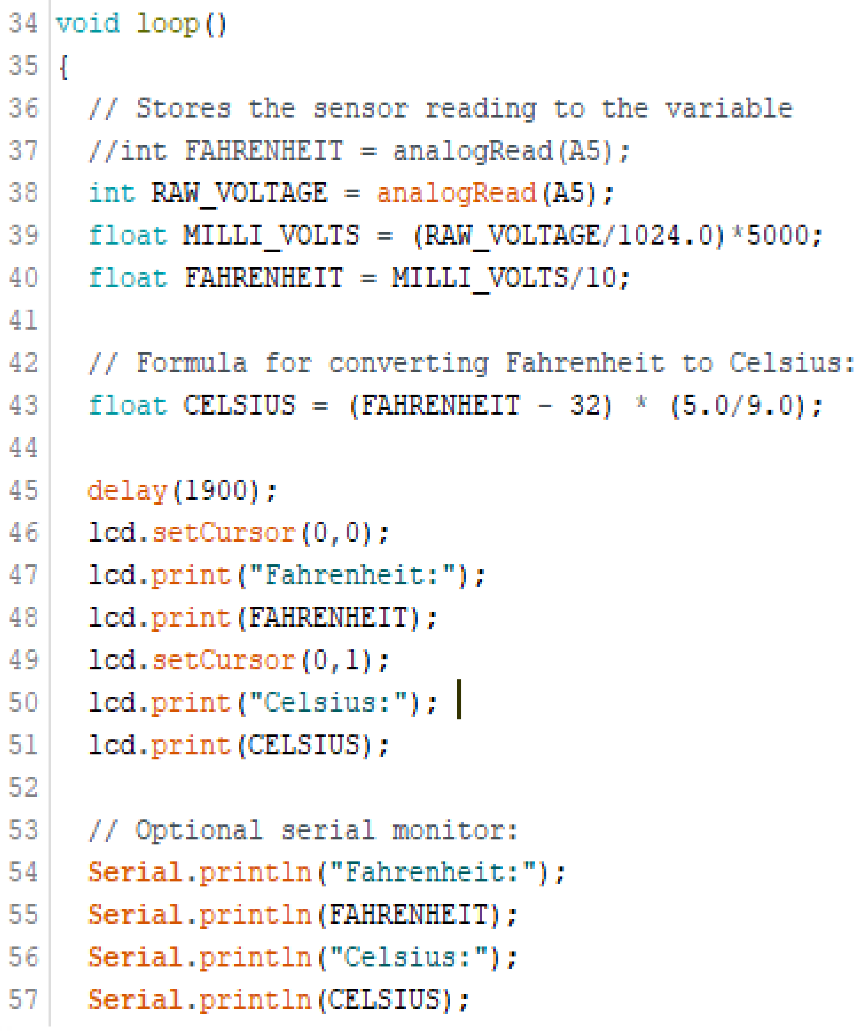


```
#include<LiquidCrystal.h>

LiquidCrystal lcd(13,12,14,15,16,17);

void setup()
{
  Serial.begin(9600); // Serial communication for checking
  lcd.begin(16,2);    // Sets LCD rows and columns
  lcd.setCursor(0,0);
  lcd.print("  TEMPERATURE");
  lcd.setCursor(0,1);
  lcd.print("    MONITOR");
  delay(1800);
  lcd.clear();
}

```

- Library used
- lcd pin assignment
  (see the wiring)

- setup
- set baudrate to 9600
- lcd display

Sample sketch for LM34 (loop)



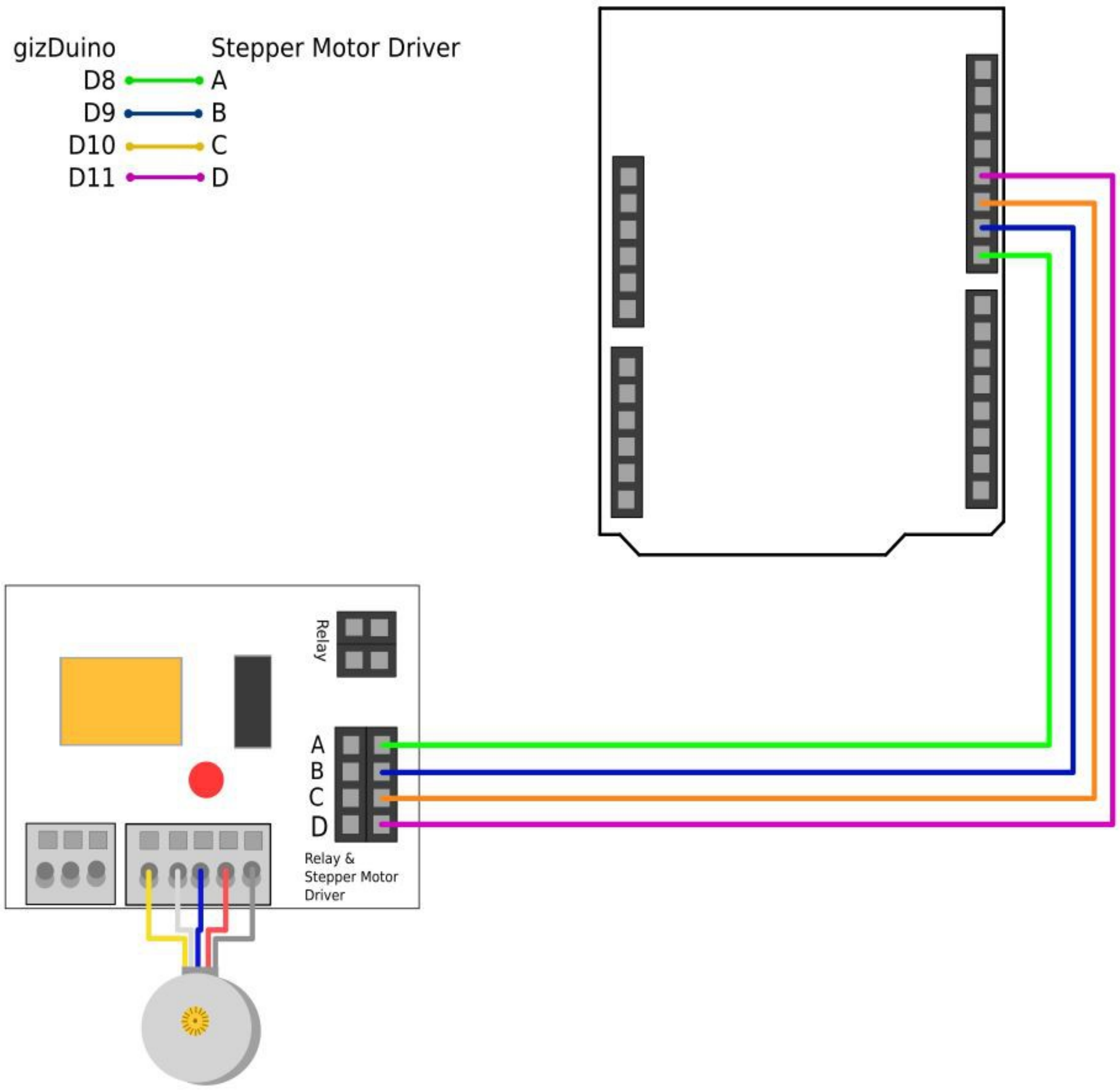
```
void loop()
{
  // Stores the sensor reading to the variable
  //int FAHRENHEIT = analogRead(A5);
  int RAW_VOLTAGE = analogRead(A5);
  float MILLI_VOLTS = (RAW_VOLTAGE/1024.0)*5000;
  float FAHRENHEIT = MILLI_VOLTS/10;

  // Formula for converting Fahrenheit to Celsius:
  float CELSIUS = (FAHRENHEIT - 32) * (5.0/9.0);

  delay(1900);
  lcd.setCursor(0,0);
  lcd.print("Fahrenheit:");
  lcd.print(FAHRENHEIT);
  lcd.setCursor(0,1);
  lcd.print("Celsius:");
  lcd.print(CELSIUS);

  // Optional serial monitor:
  Serial.println("Fahrenheit:");
  Serial.println(FAHRENHEIT);
  Serial.println("Celsius:");
  Serial.println(CELSIUS);
```

- loop
- read the analog pin A5
- convert to millivolts
- and formula to get the Fahrenheit
- and formula for Celsius

- lcd display

- serial display

Stepper Driver ULN2003A with unipolar stepper Motor

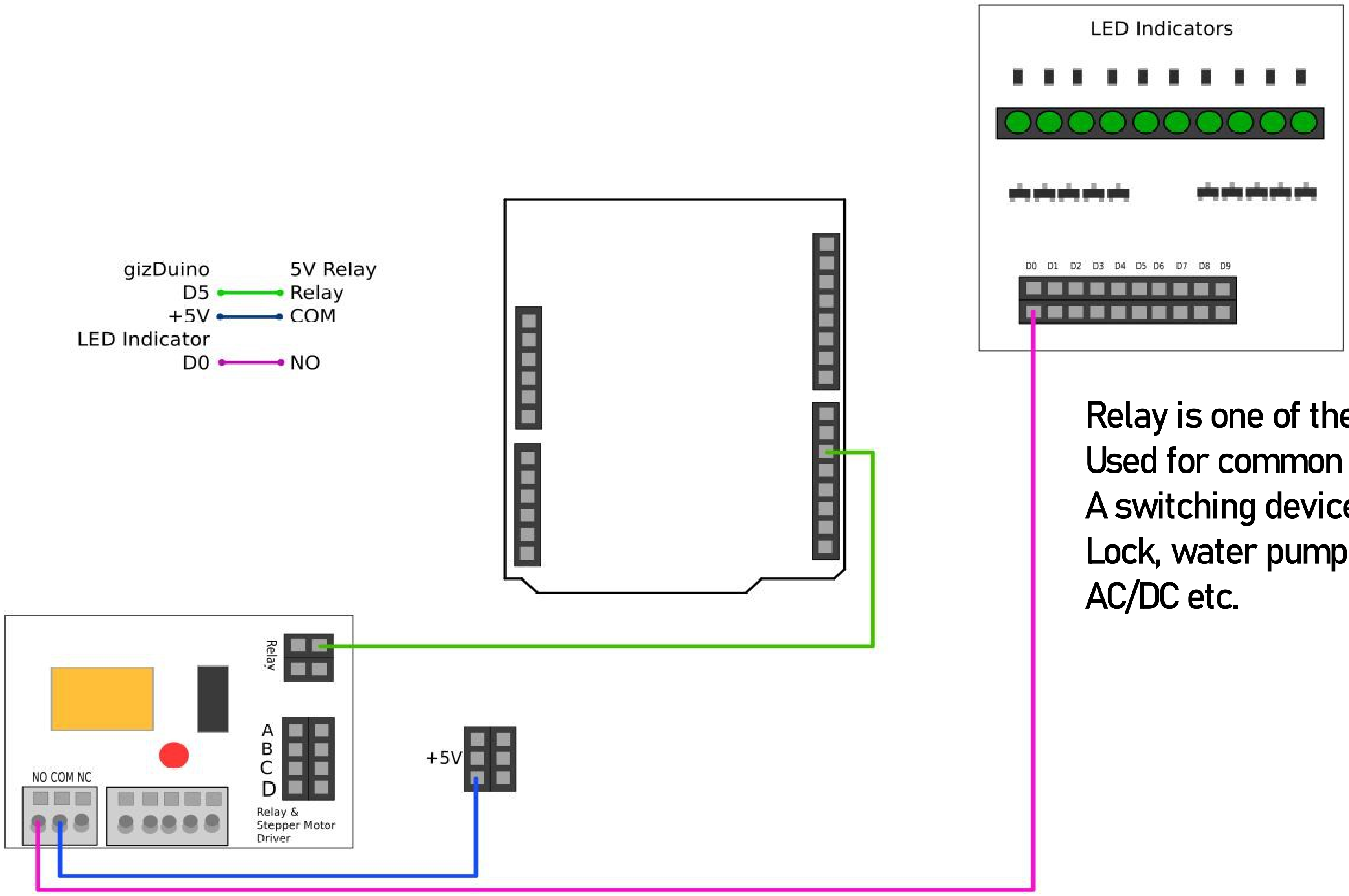


Controlling the speed and Direction of stepper motor Using the UNL2003A driver.

Sample sketch for Stepper Driver

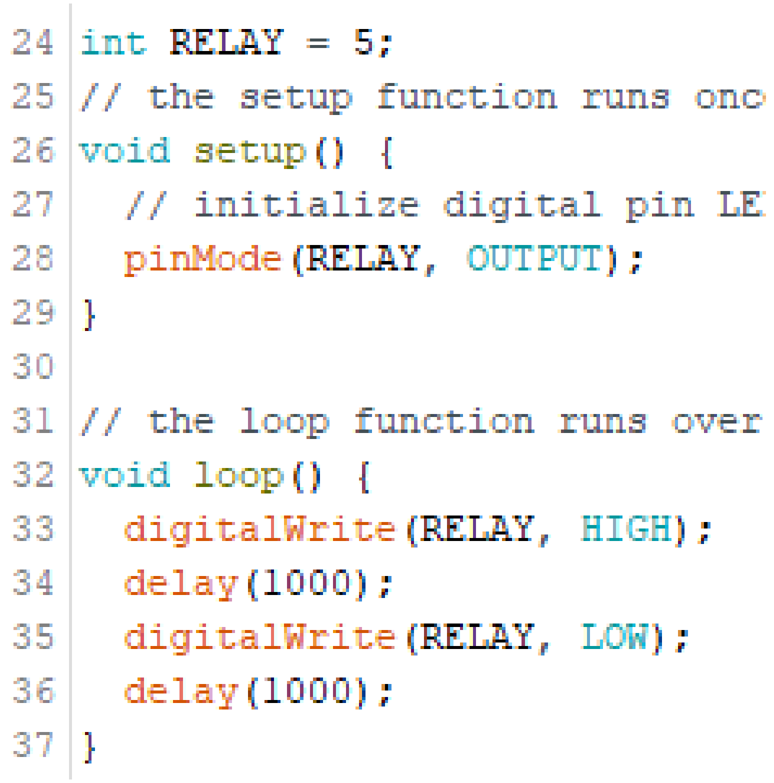
- to follow

5V Relay



Relay is one of the most
Used for common application as
A switching device for Solenoid
Lock, water pump, lights
AC/DC etc.

Sample sketch for Relay



```
int RELAY = 5;
// the setup function runs onc
void setup() {
  // initialize digital pin LE
  pinMode(RELAY, OUTPUT);
}

// the loop function runs over
void loop() {
  digitalWrite(RELAY, HIGH);
  delay(1000);
  digitalWrite(RELAY, LOW);
  delay(1000);
}
```

- Setup
Pin assignment for relay input supply.

And set it as output

- loop
  Use digitalWrite to trigger the relay.
If HIGH = relay will trigger and the COM and NO contact is connected.
If LOW = no power input. So that COM and NO contact is open.