## O-LED



/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program blinks the 2 leds on the Brainbox Arduino with 300msec intervals

-the BLUE LED at D13

-the RED LED at D17

\*/

// these lines make it possible to use pin names instead of pin numbers

// Constants do not change during the program

const int LED\_BLUE **=** 13**;**

const int LED\_RED **=** 17**;**

// this setup function runs once when you press reset or power the board

void setup**()** **{**

// initialize digital pins 13 and 17 as outputs.

pinMode**(**LED\_BLUE**,** OUTPUT**);**

pinMode**(**LED\_RED**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()** **{**

digitalWrite**(**LED\_BLUE**,** HIGH**);** // turn the BLUE LED on

digitalWrite**(**LED\_RED**,** LOW**);** // turn the RED LED off

delay**(**300**);** // wait for 300 msec

digitalWrite**(**LED\_BLUE**,** LOW**);** // turn the BLUE LED off

digitalWrite**(**LED\_RED**,** HIGH**);** // turn the RED LED on

delay**(**300**);** // wait for 300 msec

**}**

## O-Buzzer V1 met Delay



/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program blinks the 2 leds on the Brainbox Arduino with 300msec intervals

-the BLUE LED at D13

-the RED LED at D17

\*/

// these lines make it possible to use pin names in stead of pin numbers

// Constants do not change during the program

const int Buzzer **=** 7**;**

// this setup function runs once when you press reset or power the board

void setup**()** **{**

// initialize digital pin Buzzer as output

pinMode**(**Buzzer**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()** **{**

digitalWrite**(**Buzzer**,** HIGH**);** // make signal High for 1 msec

delay**(**1**);** // wait for 1 msec

digitalWrite**(**Buzzer**,** LOW**);** // make signal Low for 1 msec

delay**(**1**);** // wait for 1 msec

**}**

## O-Buzzer Siren with delay

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program lets the buzzer generate a siren

that alternates between a tone of 500Hz and a tone of 1KHz

Be aware that the 1KHz loop is looped 500 times and that the 500Hz loop is looped 250 times

to generate 500msec of the high tone and 500msec for the low tone

The buzzer is connected at IDE pin D7

A square wave that is high for 1msec and low for 1msec replicates a frequency of 500Hz.

\*/

// these lines make it possible to use pin names in stead of pin numbers

// Constants do not change during the program

const int Buzzer **=** 7**;**

// this setup function runs once when you press reset or power the board

void setup**()** **{**

// initialize digital pin Buzzer as output

pinMode**(**Buzzer**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()**

**{**

**for(** int x **=** 250**;** x**>**0 **;** x **=** x**-**1**)** //repeat this loop 250 times

**{**

digitalWrite**(**Buzzer**,** HIGH**);** // make signal High

delayMicroseconds**(**1000**);** // wait for 1 msec

digitalWrite**(**Buzzer**,** LOW**);** // make signal Low

delayMicroseconds**(**1000**);** // wait for 1 msec

**}**

**for(** int x **=** 500**;** x**>**0 **;** x **=** x**-**1**)** //repeat this loop 500 times

**{**

digitalWrite**(**Buzzer**,** HIGH**);** // make signal High

delayMicroseconds**(**500**);** // wait for 500usec

digitalWrite**(**Buzzer**,** LOW**);** // make signal Low

delayMicroseconds**(**500**);** // wait for 500usec

**}**

**}**

## O-buzzer siren with tone

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program lets the buzzer generate a siren

that alternates between a tone of 500Hz and a tone of 1KHz

We now use the tone instruction of the ARduino IDE library

tone(pin, frequency)

tone(pin, frequency, duration)

Parameters:

pin: the pin on which to generate the tone

frequency: the frequency of the tone in hertz - unsigned int

duration: the duration of the tone in milliseconds (optional) - unsigned long

The buzzer is connected at IDE pin D7

A square wave that is high for 1msec and low for 1msec replicates a frequency of 500Hz.

\*/

// these lines make it possible to use pin names in stead of pin numbers

// Constants do not change during the program

const int Buzzer **=** 7**;**

// this setup function runs once when you press reset or power the board

void setup**()** **{**

// initialize digital pin Buzzer as output

pinMode**(**Buzzer**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()**

**{**

tone**(**Buzzer**,**500**,**500**);** // on Buzzer pin - generate 500Hz signal - for 500msec

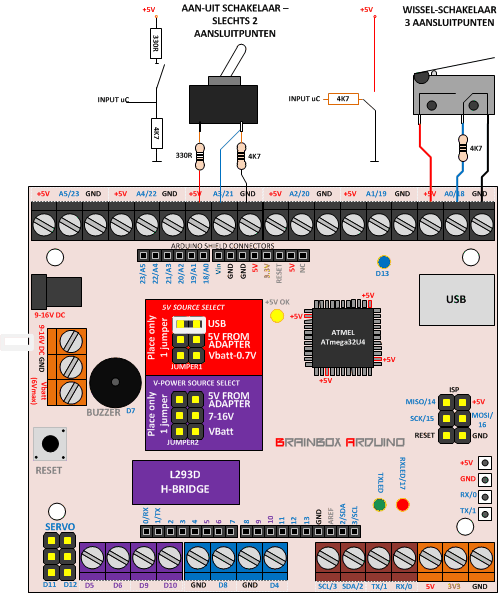
delay**(**500**);** // wait for 500msec

tone**(**Buzzer**,**1000**,**500**);**// on Buzzer pin - generate 1000Hz signal - for 500msec

delay**(**500**);** // wait for 500msec

**}**

## I-DIG

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program configures one IO pin (18 in this example) as an INPUT pin

and reads the state (1 or 0) in a variable "VAR\_IN"

If the input is high - the blue led at pin 13 will be high

If the input is low - the blue led at pin 13 will be low

Look at the Brainbox ARduino PINOUT diagram.

All the pins with a BLUE marking can be used as digital inputs

(0, 1, 2, 3, 4, 7, 8, 11, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23)

!!the 4 power output pins CAN NOT BE USED as digital inputs- (5, 6, 9, 10)

\*/

// these lines make it possible to use pin names instead of pin numbers

// Constants do not change during the program

const int IN\_PIN **=** 18**;**

const int BLUE\_LED **=** 13**;**

//this variable named VAR\_IN is used to store the state of the input pin

char VAR\_IN **=** 0**;**

// this setup function runs once when you press reset or power the board

void setup**()**

**{**

// initialize digital pin as input or output

pinMode**(**IN\_PIN**,** INPUT**);**

pinMode**(**BLUE\_LED**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()**

**{**

// read the state of the input pin :

VAR\_IN **=** digitalRead**(**IN\_PIN**);**

// if the input pin is high

**if** **(**VAR\_IN **==** HIGH**)** **{**

// turn LED on:

digitalWrite**(**BLUE\_LED**,** HIGH**);**

**}**

**else** **{** // if the input pin is low

// turn LED off:

digitalWrite**(**BLUE\_LED**,** LOW**);**

**}**

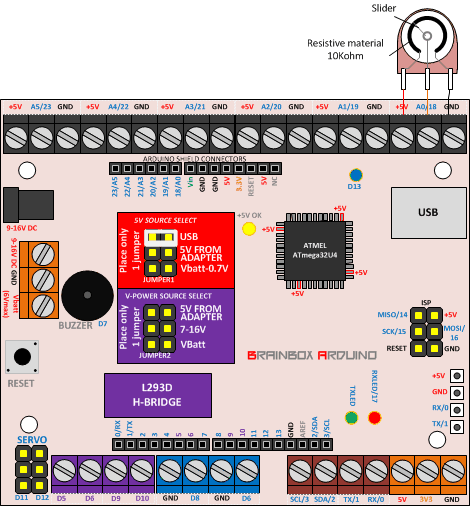
**}**

## I-AN

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016



This program is used to demonstrate how Analog inputs work with an ARduino platform

An ARduino uC is a normal AVR microcontroller but is not programmed that way.

The absense of port programming makes that we can't output a certain analog value to 8 leds or so.

The microcontroller will read the analog voltage at pin A0 and

will convert it into a digital 10 bit value between 0-1024

What we can do with an ARduino to somehow visualise this analog value is to:

- output the analog value as a PWM (AnalogWrite) to a led -> we use the BLUE led at pin 13

- Convert the Analog value into a hearable sound and use the buzzer in pin 7 for this

- Send this analog Value via the USB cable back to our computer and use the Serial monitor

of the ARduino IDE to visualise this value

We will demonstrate these 3 methods in this program

The circuit:

\* potentiometer connected to analog pin 0.

Center pin of the potentiometer goes to the analog pin.

side pins of the potentiometer go to +5V and ground

\* BLUE LED connected from digital pin 13

\* Buzzer connected to pin 7

\*/

// These constants won't change. They're used to give names

// to the pins used:

const int analogInPin **=** A0**;** // Analog input pin that the potentiometer is attached to

const int analogOutPin **=** 13**;** // Analog output pin that the BLUE LED is attached to

const int BuzzerPin **=** 7**;** // the buzzer on the Brainbox ARduino is connected to pin 7

int sensorValue **=** 0**;** // value read from the analog input - set to 0 to start

int outputValue **=** 0**;** // value output to the PWM (analog out)- set to 0 to start

void setup**()** **{**

// initialize serial communications at 9600 bps:

Serial**.**begin**(**9600**);**

**}**

void loop**()** **{**

// read the analog in value:

sensorValue **=** analogRead**(**analogInPin**);**

// map it to the range of the analog out:

// sensorvalue is a value between 0 and 1024 (10 bit AD)

// PWM or ANalogWrite can only work with values bewteen 0 and 255 (8bit)

outputValue **=** map**(**sensorValue**,** 0**,** 1023**,** 0**,** 255**);**

// change the analog out value:

// generates a PWM signal on a pin with a duty cycle between 0-255 (0-100%)

analogWrite**(**analogOutPin**,** outputValue**);**

// print the results to the serial monitor:

// Open the serial monitor of the arduino IDE to see the result

Serial**.**print**(**"sensor = " **);**

Serial**.**print**(**sensorValue**);**

Serial**.**print**(**"\t output = "**);**

Serial**.**println**(**outputValue**);**

// the buzzer converts the sensorvalue into hearable

// frequencies. The sensorValue is a value between 0-1024

// but the lowest frequency that is recognizable as a tone is

// +/- 20Hz. That is why we add 20Hz teh SensorValue.

tone **(**BuzzerPin**,** sensorValue**+**20**);**

// wait 2 milliseconds before the next loop

// for the analog-to-digital converter to settle

// after the last reading:

delay**(**2**);**

**}**

## O-20 = O-500 = O-POWER

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

04/01/2016

This program configures one IO pin (D4 in this example) as an output pin and

alternates between making this pin high or low with 1 sec delays

It can be adapted to drive leds - RGB leds or any device that does not draw current over 20mA

Look at the Brainbox ARduino PINOUT diagram. ALl the pins with a blue marking can be used!

It can be altered to be used at any IO pin available at the Brainbox ARduino (max current 20mA)

(0, 1, 2, 3, 4, 7, 8, 11, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23)

It can also be used at the 4 power output pins - (5, 6, 9, 10) (max current 600mA)

(pins with a purple marking at PINOUT diagram)

Be aware that the voltage at these 4 pins needs to be set by jumper 2

\*/

// these lines make it possible to use pin names instead of pin numbers

// Constants do not change during the program

const int OUT\_PIN **=** 4**;**

// this setup function runs once when you press reset or power the board

void setup**()**

**{**

// initialize digital pin as output

pinMode**(**OUT\_PIN**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()**

**{**

digitalWrite**(**OUT\_PIN**,** HIGH**);** // make pin high

delay**(**1000**);** // wait for 1sec

digitalWrite**(**OUT\_PIN**,** LOW**);** // on Buzzer pin - generate 500Hz signal - for 500msec

delay**(**1000**);** // make pin low

**}**

## O-STEPPER without function

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

13/01/2016

This program configires the 4 power output pins (D5, D6, D9, D10) to drive a stepper motor

Connect Phase 1 of the stepper motor between D5 and D6

Connect Phase 2 of the stepper motor between D9 and D10

Look at the worksheet 0-500 on how stepper motors should be connected

Be aware that the maximum output current of 600mA for these 4 power output pins may not be exceeded.

This program makes the stepper motor turn 40x4 steps forward and then 40x4 steps backwards

Be aware that the voltage at these 4 power output pins needs to be set with jumper 2

\*/

// these lines make it possible to use pin names instead of pin numbers

// Constants do not change during the program

const int Ph1\_Pin1 **=** 5**;**

const int Ph1\_Pin2 **=** 6**;**

const int Ph2\_Pin1 **=** 9**;**

const int Ph2\_Pin2 **=** 10**;**

// this setup function runs once when you press reset or power the board

void setup**()**

**{**

// initialize digital pin as output

pinMode**(**Ph1\_Pin1**,** OUTPUT**);**

pinMode**(**Ph1\_Pin2**,** OUTPUT**);**

pinMode**(**Ph2\_Pin1**,** OUTPUT**);**

pinMode**(**Ph2\_Pin2**,** OUTPUT**);**

**}**

// this loop function runs over and over again forever

void loop**()**

**{**

**while(**1**)**

**{**

**for** **(**char x **=** 40**;** x**>**0**;** x**--)** // 40 steps forward - WAVE STEP

**{**

digitalWrite**(**Ph1\_Pin1**,** HIGH**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 1 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** HIGH**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 2 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** HIGH**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 3 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** HIGH**);** //sequence 4 of 4

delay**(**20**);** // wait for ..msec

**}**

**for** **(**char x **=** 40**;** x**>**0**;** x**--)** // 40 steps backward - WAVE STEP

**{**

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** HIGH**);** //sequence 4 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** HIGH**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 3 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** LOW**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** HIGH**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 2 of 4

delay**(**20**);** // wait for ..msec

digitalWrite**(**Ph1\_Pin1**,** HIGH**);** digitalWrite**(**Ph1\_Pin2**,** LOW**);** digitalWrite**(**Ph2\_Pin1**,** LOW**);** digitalWrite**(**Ph2\_Pin2**,** LOW**);** //sequence 1 of 4

delay**(**20**);** // wait for ..msec

**}**

**}**

**}**

## O-STEPPER with function

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

13/01/2016

This program configires the 4 power output pins (D5, D6, D9, D10) to drive a stepper motor

We will drive all the 4 pins with the usage of the stepper motor driver function of ARduino

Connect Phase 1 of the stepper motor between D5 and D6

Connect Phase 2 of the stepper motor between D9 and D10

Look at the worksheet 0-500 STEPPER on how stepper motors should be connected

Be aware that the maximum output current of 600mA for these 4 power output pins may not be exceeded.

This program makes the stepper motor turn 40x4 steps forward and then 40x4 steps backwards

Be aware that the voltage at these 4 power output pins needs to be set with jumper 2

\*/

#include <Stepper.h>

const int stepsPerRevolution **=** 100**;** // change this to fit the number of steps per revolution

// for your motor

// initialize the stepper library on pins 8 through 11:

Stepper myStepper**(**stepsPerRevolution**,** 5**,** 6**,** 9**,** 10**);**

void setup**()**

**{**

// set the speed at 60 rpm:

myStepper**.**setSpeed**(**60**);**

**}**

void loop**()** **{**

// step one revolution in one direction:

myStepper**.**step**(**stepsPerRevolution**);**

delay**(**500**);**

// step one revolution in the other direction:

myStepper**.**step**(-**stepsPerRevolution**);**

delay**(**500**);**

**}**

## O-PWM

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

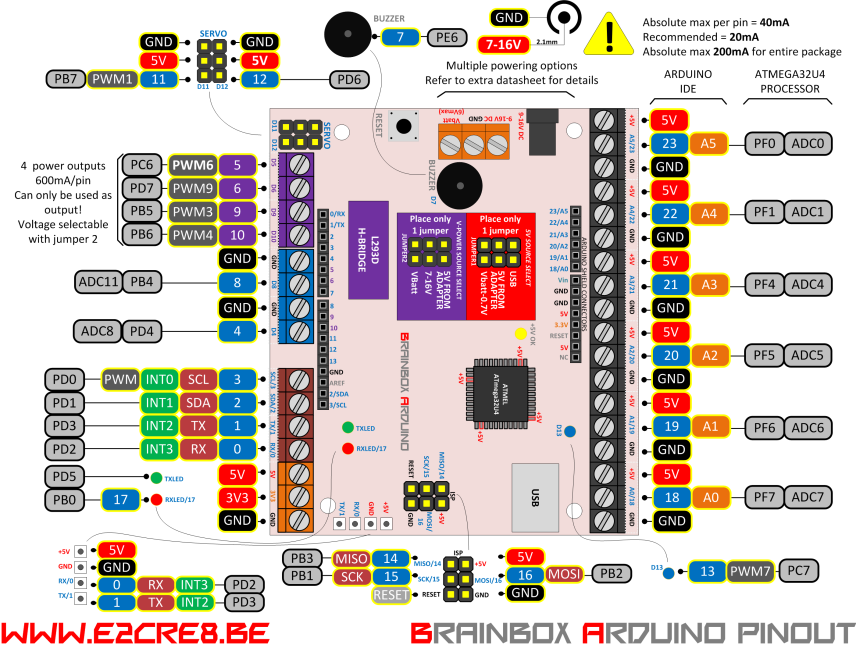
13/01/2016

This program generates a PWM signal on one of the pins that have PWM functionality

The program generates a PWM signal with this sequence:

1- Rising PWM duty cycle from 0-100% (0-255) over a period of +/-5sec

2- Falling PWM duty cycle from 100% to 0% (255-0) over a period of +/- 5sec

In this case we use pin D5. You could connect a DC motor (600mA max) between D5 and GND

You could also test this with a led (with resistor) between D5 and GND

Be aware that the 600mA power outputs need a jumper to select the output voltage

The pins on the BBA that have PWM functionality are:

D3 20mA max

D5 600mA max

D6 600mA max

D9 600mA max

D10 600mA max

D11 20mA max

D13 20mA max (Blue led)

\*/

// these lines make it possible to use pin names instead of pin numbers

// Constants do not change during the program

const int PWM\_PIN **=** 5**;**

// this setup function runs once when you press reset or power the board

void setup**()**

**{**

pinMode**(**PWM\_PIN**,** OUTPUT**);** // initialize digital pin as output

**}**

void loop**()**// this loop function runs over and over again forever

**{**

**while(**1**)**

**{**

**for** **(**int brightness **=** 0**;** brightness**<**255**;** brightness**++)** // loop counter 0 up to 255

**{**

analogWrite**(**PWM\_PIN**,** brightness**);** // PWM output signal

delay**(**20**);** // wait for ..msec (255x20msec = 5.1sec)

**}**

**for** **(**int brightness **=** 255**;** brightness**>**0**;** brightness**--)** // loop counter 255 down to 0

**{**

analogWrite**(**PWM\_PIN**,** brightness**);** // PWM output signal

delay**(**20**);** // wait for ..msec (255x20msec = 5.1sec)

**}**

**}**

**}**

## O-SERVO

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

13/01/2016

This program generates a SERVO signal on pin D11 and D12

up to 12 servo signals can be generated on any other IO pin of the Brainbox Arduino, but D11&D12 already have standard servo connectors

To generate this specific servo signal we make use of the Servo library of Arduino

Most servo's work with signals between 1000msec and 2000msec. That is exactly what the "myservo.write(pos)" instruction does.

Note that some manufactures do not follow this standard very closely so that servos often respond to values between 700 and 2300.

Feel free to use the "servo.writeMicroseconds(uS)" instruction to increase these endpoints until the servo no longer continues to increase its range. \*/

#include <Servo.h>

Servo Servo\_D11**;** // create servo object to control a servo

Servo Servo\_D12**;** // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos **=** 0**;** // variable to store the servo position

void setup**()**

**{**

Servo\_D11**.**attach**(**11**);** // attaches the servo on pin 11 to the servo object

Servo\_D12**.**attach**(**12**);** // attaches the servo on pin 12 to the servo object

**}**

void loop**()**

**{**

**for(**pos **=** 0**;** pos **<=** 180**;** pos**++)** // goes from 0 degrees to 180 degrees

**{** // in steps of 1 degree

Servo\_D11**.**write**(**pos**);** // tell servo to go to position in variable 'pos'

Servo\_D12**.**write**(**pos**);** // tell servo to go to position in variable 'pos'

delay**(**15**);** // waits 15ms for the servo to reach the position

**}**

**for(**pos **=** 180**;** pos**>=**0**;** pos**--)** // goes from 180 degrees to 0 degrees

**{**

Servo\_D11**.**write**(**pos**);** // tell servo to go to position in variable 'pos'

Servo\_D12**.**write**(**pos**);** // tell servo to go to position in variable 'pos'

delay**(**15**);** // waits 15ms for the servo to reach the position

**}**

**}**

## I2C-LCD



/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

13/01/2016

This program drives an I2C LCD of the type:

16 character 2 line I2C Display

Backpack Interface labelled "YwRobot Arduino LCM1602 IIC V1" (2€ @ aliexpress)

Connect this LCD as follows:

LCD Brainbox Arduino

GND GND

VCC +5V

SDA SDA/2

SCL SCL/3

!! Pull up resistors are required - place 4K7 between SDA and 5V and 4K7 between SCL and 5V

To communicate correctly with this I2C LCD you need to install the <LiquidCrystal\_I2C.h> library in the arduino IDE

1- download the "LiquidCrystal\_I2C" library as a zip file from https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads

2- do not unzip the file

3- in ARduino IDE: Sketch >> include library >> add .ZIP library - select the downloaded zip file

4- this library is installed under 'mydocs'->Arduino : remove it by deleting it there

\*/

#include <Wire.h> // Comes with Arduino IDE

// Get the LCD I2C Library here:

// https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads

// Move any other LCD libraries to another folder or delete them

// See Library "Docs" folder for possible commands etc.

#include <LiquidCrystal\_I2C.h>

const int analogInPin **=** A0**;** // Analog input pin that the potentiometer is attached to

int sensorValue **=** 0**;**

int outputValue **=** 0**;**

// set the LCD address to 0x27 for a 20 chars 4 line display

// Set the pins on the I2C chip used for LCD connections:

// addr, en,rw,rs,d4,d5,d6,d7,bl,blpol

LiquidCrystal\_I2C lcd**(**0x27**,** 2**,** 1**,** 0**,** 4**,** 5**,** 6**,** 7**,** 3**,** POSITIVE**);** // Set the LCD I2C address

void setup**()**

**{**

lcd**.**begin**(**16**,**2**);**// initialize library

**for(**int i **=** 0**;** i**<** 3**;** i**++)** // loop 3 times backlight on and off

**{**

lcd**.**backlight**();** //backlight on

delay**(**250**);**

lcd**.**noBacklight**();** //backlight off

delay**(**250**);**

**}**

lcd**.**backlight**();** //backlight on

lcd**.**setCursor**(**0**,**0**);** // set cursor to positon x=0, y=0

lcd**.**print**(**"Brainbox"**);** // print text on the LCD

delay**(**500**);**

lcd**.**setCursor**(**2**,**1**);**

lcd**.**print**(**"Arduino"**);**

delay**(**1000**);**

**}**

void loop**()**

**{**

sensorValue **=** analogRead**(**analogInPin**);** // read the analog value measured at analog pin AN0 (pin18) - We used a potmeter

outputValue **=** map**(**sensorValue**,** 0**,** 1023**,** 0**,** 100**);** // rescale the sensorValue (0-1024) to (0-100)%

lcd**.**clear**();** // clear the LCD

lcd**.**setCursor**(**0**,**0**);** // set cursor to positon x,y

lcd**.**print**(**"AN0 Value:"**);** // print text on the LCD

lcd**.**print**(**sensorValue**);** // print the value of the variable on the LCD

lcd**.**setCursor**(**0**,**1**);**

lcd**.**print**(**"AN0 %:"**);**

lcd**.**print**(**outputValue**);**

lcd**.**print**(**"%"**);** // add a % character

delay**(**100**);** // delay of 100msec to avoid flikkering of the LCD

**}**

## USB Serial monitor

/\*

www.E2CRE8.be - Brainbox Arduino - by Bart Huyskens

20/02/2016

6 CHANNEL Analog input, serial output

Reads an analog input pin, maps the result to a range from 0 to 255

and prints the results to the serial monitor.

Use the serial monitor built into ARduino IDE to visualise the results

6 analog input channels - A0-A5

Be aware that the serial monitor uses the same USB port as IDE uses to program tne Leonardo

Close the serial monitor program and reset the Leonardo to set it up to receive new programs

\*/

// These constants won't change. They're used to give names

// to the pins used:

const int analogInPin **=** A0**;** // Analog input pin that the potentiometer is attached to

const int analogOutPin **=** 9**;** // Analog output pin that the LED is attached to

int sensorValue **=** 0**;** // value read from the pot

int outputValue **=** 0**;** // value output to the PWM (analog out)

//Declare the AD variables as int because Arduino always executes a 10 bit AD conversion

int AD\_A0 **=** 0**;**

int AD\_A1 **=** 0**;**

int AD\_A2 **=** 0**;**

int AD\_A3 **=** 0**;**

int AD\_A4 **=** 0**;**

int AD\_A5 **=** 0**;**

// These 8 bit Byte variables are used to downsize the 10 bit AD conversion result into 8 bit

byte ADC\_A0 **=** 0**;**

byte ADC\_A1 **=** 0**;**

byte ADC\_A2 **=** 0**;**

byte ADC\_A3 **=** 0**;**

byte ADC\_A4 **=** 0**;**

byte ADC\_A5 **=** 0**;**

void setup**()** **{**

// initialize serial communications at 9600 bps via USB

Serial**.**begin**(**9600**);**

**}**

void loop**()** **{**

AD\_A0 **=** analogRead**(**A0**);** // read the analog value measured at analog pin ANx

AD\_A1 **=** analogRead**(**A1**);** // read the analog value measured at analog pin ANx

AD\_A2 **=** analogRead**(**A2**);** // read the analog value measured at analog pin ANx

AD\_A3 **=** analogRead**(**A3**);** // read the analog value measured at analog pin ANx

AD\_A4 **=** analogRead**(**A4**);** // read the analog value measured at analog pin ANx

AD\_A5 **=** analogRead**(**A5**);** // read the analog value measured at analog pin ANx

ADC\_A0 **=** map**(**AD\_A0**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

ADC\_A1 **=** map**(**AD\_A1**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

ADC\_A2 **=** map**(**AD\_A2**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

ADC\_A3 **=** map**(**AD\_A3**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

ADC\_A4 **=** map**(**AD\_A4**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

ADC\_A5 **=** map**(**AD\_A5**,** 0**,** 1023**,** 0**,** 255**);** // rescale the sensorValue (0-1024) to (0-255)

// print the results to the serial monitor:

Serial**.**print**(**"A0 = "**);**

Serial**.**print**(**ADC\_A0**,** DEC**);**

Serial**.**print**(**"\t"**);**

Serial**.**print**(**"A1 = "**);**

Serial**.**print**(**ADC\_A1**,** HEX**);**

Serial**.**print**(**"\t"**);**

Serial**.**print**(**"A2 = "**);**

Serial**.**print**(**ADC\_A2**,** BIN**);**

Serial**.**print**(**"\t"**);**

Serial**.**print**(**"A3 = "**);**

Serial**.**print**(**ADC\_A3**);**

Serial**.**print**(**"\t"**);**

Serial**.**print**(**"A4 = "**);**

Serial**.**print**(**ADC\_A4**);**

Serial**.**print**(**"\t"**);**

Serial**.**print**(**"A5 = "**);**

Serial**.**print**(**ADC\_A5**);**

Serial**.**println**(**""**);** // start new line

delay**(**200**);**

**}**