Arduino Workshop

Duration: 2 hours

In this workshop, we will get familiar with the Arduino culture and interfaces, and try to work with as much electrical components as possible.

Participants are advised to split themselves in groups of 3 (depending on the number of available boards). The audience attending the workshop has different knowledge levels and skills, some have experience, some other haven't even heard of Arduino before, that's why it is highly recommended that the team incorporates at least an experienced member in order to help his team mates and get work done faster, especially that the time limit is merely 2 hours

The workshop consists of a series of advanced projects to by each team; Around 12 applications will be held during this workshop, but since time is too short, they will take place in parallel, for example among 12 possible projects, a team will develop only one project of his choice, so each group can choose the projects that best satisfy their needs and expectations. If a group finished with a project within the time limit, they can start developing another project.

Tools for download: www.bit.ly/twesdArduino

List of projects:

In this section, we will control several new components in the form of simple projects while learning the basics of Arduino and electrical connections.

As mentioned before, the team should have at least a member that is familiar with Arduino

Project 1: 7-segments display

The project consists of using a 7-segments display to display a random number between 0 and 9; each second

1 team is involved in this project, the number of components available is enough for **2** teams

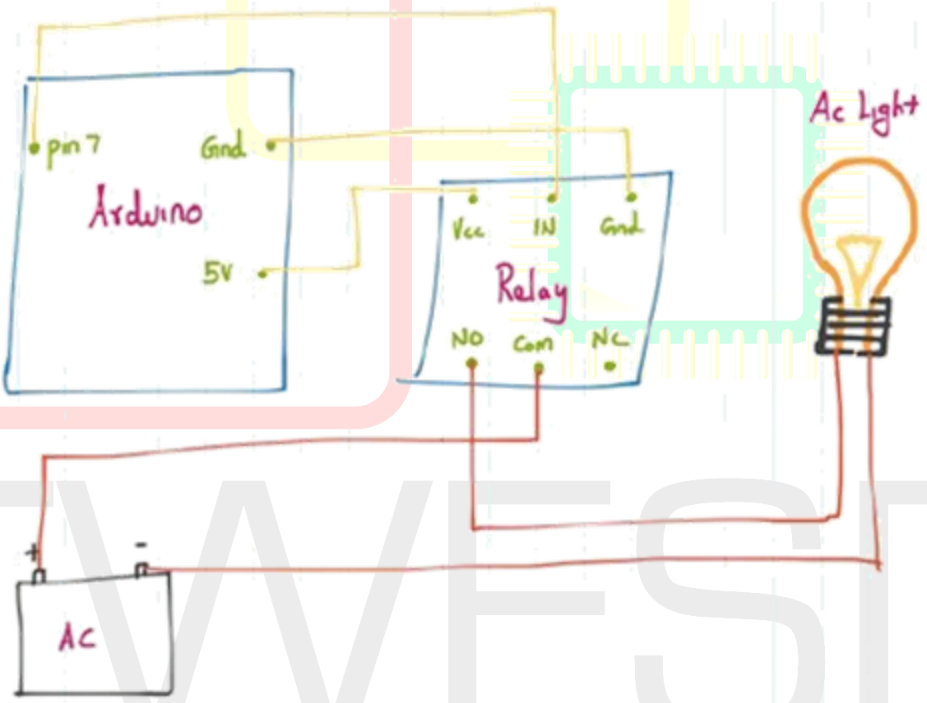
Project 2: delayed relay

This project consists of having a relay activated after a certain delay, that delay is displayed in a 7 segments display

This projects involves working with 2 main components: 7-segment display & relay

1 team is involved in this project, the number of available components is enough for **3** teams

Relay wiring:



Project 3: timer

The project consists of developing a decrementing counter (the maximum value is 9999), the remaining time is displayed using a 7-segments display with 4 digits. The challenge here is display the 4 digits simultaneously and having the time update itself each second with interrupting Arduino's loop work

Another version of this project is to use 3 separate 7-segments display and develop a counting timer with 999 as max value

1 team is involved in each one of these **2** projects

Project 4: Servo motor

This projects consists of using a potentiometer to control the rotation angle of a servo motor

1 team is involved in this project, the number of components available is enough for **3** teams

Project 5: LCD screen

This project consists of using an LCD screen to display the distance measured by an ultrasound sensor

1 team is involved in this project

LCD screen code snippet:

```
//* LCD RS pin to digital pin 12
//* LCD Enable pin to digital pin 11
//* LCD D4 pin to digital pin 5
//* LCD D5 pin to digital pin 4
//* LCD D6 pin to digital pin 3
//* LCD D7 pin to digital pin 2
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // initialize the library with the
numbers of the interface pins

void setup() {
  lcd.begin(16, 2);   // set up the LCD's number of columns and rows:
  lcd.print("hello, world!");   // Print a message to the LCD.
}

void loop() {
  lcd.setCursor(0, 1);   // set the cursor to column 1, line 2 (counting
start from 0)
  lcd.print(millis() / 1000);   // print the number of seconds since reset:
}
```

Project 6: motor control

In this project, we will control a motor.

When talking about motor, we talk about high voltage & high current, values that are way too high for the Arduino board to handle and risk of damage is very high. In order to control a motor via Arduino, we need to motor driver to amplify the voltage and power the motor using simple Arduino signals, the drive ensure also the protection of the board from current overload coming back from the motor

In this workshop, we will use 2 drivers: L298 & L9110

2 teams are involved in this project, each one will be using a driver

Project 7: Sensor data logger

In this project we will read the values from a sensor of your choice and these data will be stored inside a micro SD card.

In this workshop, there are several ways to connect Arduino to the SD card:

- Dedicated SD module
- Ethernet shield
- TFT shield
- Wireless SD shield

Note: for additional code samples, please refer to the Arduino library

1 team is involved in this project, the number of available components is enough for **4** teams

Project 8: password typing with keypad

The project consists of using a keypad component to type a passcode, if the code is correct a green LED will be on, otherwise, a RED led will stay off. The code is stored in the EEPROM of the board

This project involves **3** teams to realize 3 tasks:

- Write data to EEPROM
- Read data from the EEPROM
- Read keystrokes typed from the keypad and build the final code

EEPROM write snippet:

```
#include <EEPROM.h>
int addr = 0; // current address in the EEPROM (which byte we're going to write to next)
void setup() {}

void loop() {
  int val = analogRead(0) / 4;
  EEPROM.write(addr, val);

  addr = addr + 1;
  if (addr == EEPROM.length()) { addr = 0; }

 /***  As the EEPROM sizes are powers of two, wrapping (preventing overflow) of an EEPROM address is also doable by a bitwise and of the length - 1.
   ++addr &= EEPROM.length() - 1;  ***/
  delay(100);
}
```

EEPROM read snippet

```
#include <EEPROM.h>
int address = 0; // start reading from the first byte (address 0) of the
EEPROM

void setup() {
  Serial.begin(9600); }

void loop() {
  byte value = EEPROM.read(address);    // read a byte from the current
address of the EEPROM
  Serial.print(address);
  Serial.print("\t");
  Serial.print(value, DEC);
  Serial.println();

  address = address + 1;
  if (address == EEPROM.length()) { address = 0; }
  /***    As the EEPROM sizes are powers of two, wrapping (preventing overflow) of an EEPROM
address is also doable by a bitwise and of the length - 1.
    ++address &= EEPROM.length() - 1;    ***/
  delay(500);
}
```

Keypad code snippet:

//pins 2==>9 used, pin 2 connected to pin1 of keypad & keypad library is added

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //FOUR columns
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'} };
byte rowPins[ROWS] = {x, x, x, x}; //connect the row pins of the keypad
byte colPins[COLS] = {x, x, x, x}; //connect the column pins of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600); }

void loop(){
  char key = keypad.getKey();
  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

Project 9: exchange data wirelessly

This project consists of using an RF emitter receiver to send data from Arduino board #1 to board #2

2 programs need to be developed: a team will develop the program to send information and the other team will work on the program to receive this data

2 teams will be involved in this project, the number of available components is enough for **3** projects (**6** teams)

Emitter code snippet

```
#include <VirtualWire.h>
char *controller;
void setup() {
vw_set_ptt_inverted(true); //
vw_set_tx_pin(12);
vw_setup(4000);// speed of data transfer Kbps
}

void loop(){
controller="1"  ;
vw_send((uint8_t *)controller, strlen(controller));
vw_wait_tx(); // Wait until the whole message is gone
delay(2000);
controller="0"  ;
vw_send( (uint8_t *)controller, strlen(controller) );
vw_wait_tx();
delay(2000); }
```

Receiver code snippet

```
#include <VirtualWire.h>
void setup() {
  vw_set_ptt_inverted(true); // Required for DR3100
  vw_set_rx_pin(10);
  vw_setup(4000);  // Bits per sec
  vw_rx_start();       // Start the receiver PLL running
}
  void loop() {
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;

  if (vw_get_message(buf, &buflen)) {// Non-blocking
    if(buf[0]=='x'){   }
    if(buf[0]=='y'){   }
}}
```

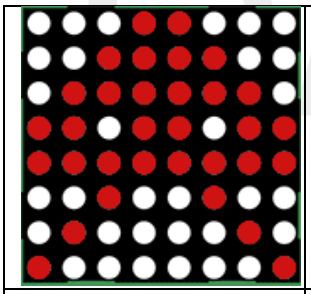
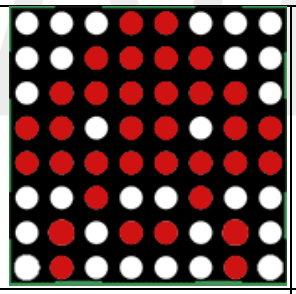
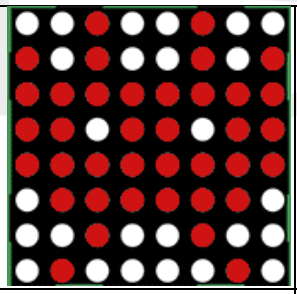
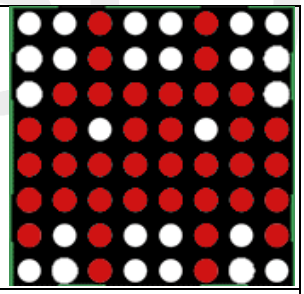
Project 10: LED matrix drawing

This project consists of displaying 2 LED matrixes to display animated tiles of space invaders

Two tiles will be drawn, for each one of them we draw two instances

1 team is involved in this project

Animated tiles to draw:

			
Tile1, frame 1	Tile1, frame 2	Tile2, frame 1	Tile2, frame 2

LED matrix code snippet:

```
#include "LedControl.h"
LedControl lc=LedControl(12,11,10,2);   // Pins: DIN,CLK,CS, # of Display connected
unsigned long delayTime=200;   // Delay between Frames

byte invader1a[] ={// First frame of invader #1
  B00011000,
};

byte invader1b[] ={// Second frame of invader #1

};

byte invader2a[] ={// First frame of invader #2

};

byte invader2b[] ={// Second frame of invader #2
};

void setup() {
  lc.shutdown(0,false);  // Wake up displays
  lc.shutdown(1,false);
  lc.setIntensity(0,5);  // Set intensity levels
  lc.setIntensity(1,5);
  lc.clearDisplay(0);  // Clear Displays
  lc.clearDisplay(1);
}

//  Take values in Arrays and Display them
void sinvader1a() { for (int i = 0; i < 8; i++) {
lc.setRow(0,i,invader1a[i]); }}
void sinvader1b() { for (int i = 0; i < 8; i++) {
lc.setRow(0,i,invader1b[i]); }}

void loop() {
    sinvader1a(); // Put #1 frame on both Display
    delay(delayTime);
    sinvader2a();
    delay(delayTime);
}
```

Project 11: interface a joystick

In this project, we will control the state of different LEDs using a joystick controller, the joystick is controlled by an Arduino board using a USB host shield

1 team is involved in this project

Joystick detection code:

```
#include <usbhid.h>
#include <hiduniversal.h>
#include <usbhub.h>
#include "hidjoystickrptparser.h"
// Satisfy IDE, which only needs to see the include statment in the ino.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#include <SPI.h>
#endif

USB Usb;
USBHub Hub(&Usb);
HIDUniversal Hid(&Usb);
JoystickEvents JoyEvents;
JoystickReportParser Joy(&JoyEvents);

void setup() {
  Serial.begin(115200);
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo,
Teensy and other boards with built-in USB CDC serial connection
#endif
        Serial.println("Start");
        if (Usb.Init() == -1)  Serial.println("OSC did not start."); else
Serial.println("OSC start.");
        delay(200);
        if (!Hid.SetReportParser(0, &Joy))  ErrorMessage<uint8_t >
(PSTR("SetReportParser"), 1);
}

void loop() { Usb.Task(); }

JoystickReportParser::JoystickReportParser(JoystickEvents *evt) :
joyEvents(evt),
oldHat(0xDE),
oldButtons(0) { for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++) oldPad[i] =
0xD; }

void JoystickReportParser::Parse(USBHID *hid, bool is_rpt_id, uint8_t len,
uint8_t *buf) {
        bool match = true;

        for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++)        // Checking if
there are changes in report since the method was last called
                if (buf[i] != oldPad[i]) {
                        match = false;
                        break;
                }

        if (!match && joyEvents) {    // Calling Game Pad event handler
                joyEvents->OnGamePadChanged((const GamePadEventData*)buf);
```

```
                for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++) oldPad[i] =
buf[i];
        }

        uint8_t hat = (buf[5] & 0xF);

        if (hat != oldHat && joyEvents) {   // Calling Hat Switch event
handler
                joyEvents->OnHatSwitch(hat);
                oldHat = hat;
        }

        uint16_t buttons = (0x0000 | buf[6]);
        buttons <<= 4;
        buttons |= (buf[5] >> 4);
        uint16_t changes = (buttons ^ oldButtons);

        if (changes) { // Calling Button Event Handler for every button
changed
                for (uint8_t i = 0; i < 0x0C; i++) {
                        uint16_t mask = (0x0001 << i);
                        if (((mask & changes) > 0) && joyEvents)
                                if ((buttons & mask) > 0) joyEvents-
>OnButtonDn(i + 1);
                                else                      joyEvents-
>OnButtonUp(i + 1);      }
                oldButtons = buttons;
        }}

void JoystickEvents::OnGamePadChanged(const GamePadEventData *evt) {
        Serial.print("X1: ");         PrintHex<uint8_t > (evt->X, 0x80);
        Serial.print("\tY1: ");       PrintHex<uint8_t > (evt->Y, 0x80);
        Serial.print("\tX2: ");       PrintHex<uint8_t > (evt->Z1, 0x80);
        Serial.print("\tY2: ");       PrintHex<uint8_t > (evt->Z2, 0x80);
        Serial.print("\tRz: ") ;       PrintHex<uint8_t > (evt->Rz, 0x80);
        Serial.println("");
        if (evt->Z1==4) Serial.println(" \n\nYES!\n ");
}

void JoystickEvents::OnHatSwitch(uint8_t hat) {
        Serial.print("Hat Switch: "); PrintHex<uint8_t > (hat, 0x80);
        Serial.println(""); }

void JoystickEvents::OnButtonUp(uint8_t but_id) {
        Serial.print("Up: ");         Serial.println(but_id, DEC);
Serial.println("test"); }

void JoystickEvents::OnButtonDn(uint8_t but_id) {
        Serial.print("Dn: ");       Serial.println(but_id, DEC); }
```

Project 12: play sound

This project consists of using a piezo sensor to generate a sound (the piezo sensor is also a sound buzzer)

1 team is involved in this project, the number of components available is enough for **2** teams

Sound sample:

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

//  melody[] is an array of notes, accompanied by beats[], which sets each
note's relative length (higher #, longer note)
int melody[] = {  1912,  2028,  2550,  1912,  2028,   3038,  0,  1912,
3830,  2550, 2272, 1912 };
int beats[]  = {   16 ,   16 ,   16 ,    8 ,    8 ,    16 , 32,   16 ,   16
,   16 ,   8 ,   8 };

long tempo = 10000; // Set overall tempo
int pause = 1000; // Set length of pause between notes
int rest_count = 100; //// Loop variable to increase Rest length <-
BLETCHEROUS HACK; See NOTES

void loop() {
  for (int i=0; i<sizeof(melody)/2; i++) { // Set up a counter to pull from
melody[] and beats[]
    int tone_ = melody[i];
    int beat = beats[i];
    long duration = beat * tempo; // Set up timing

    long elapsed_time = 0;
  if (tone_ > 0) { // if this isn't a Rest beat, while the tone has played
less long than 'duration', pulse speaker HIGH and LOW
    while (elapsed_time < duration) {
      digitalWrite(9,HIGH);
      delayMicroseconds(tone_ / 2);
      digitalWrite(9, LOW);
      delayMicroseconds(tone_ / 2);
      elapsed_time += (tone_);      // Keep track of how long we pulsed
    }}

  else for (int j = 0; j < rest_count; j++) delayMicroseconds(duration);
// Rest beat; loop times delay
  }}
```