OPEN LABS V1

I. GENERAL SPECIFICATIONS

Instrument module	Arduino based
	2 channels
	10 bit ADC resolution
	9600 baud rate
	Unchangeable 2.5V offset
	AC & DC input
	2.5V maximum amplitude
Client software	Built in Python
	HTTP request method
Web application	HTML filling
	CSS & JavaScript front end
	PHP back end
	MySQL database
Web server	2GB RAM
	1 CPU core
	2TB transfer
	24 GB SSD
	2GB RAM

II. SYSTEM REQUIREMENTS & DEPENDANCIES

Section	Sub-section	Specific
Instrument hardware	Microcontroller	Arduino UNO R3
		Firmware
	Signal conditioner circuit	2 Resistors (100k)
Client side	Hardware	Laptop/Desktop computer
		Internet connection
	Software	UBUNTU OS
		Python script
		Python Environment
Server side	Hardware	2GB RAM
		1 CPU core
		2TB transfer
		24 GB SSD
	Software	UBUNTU Server OS
		Apache Web Sever
		MySQL
		PHP compiler

III. FUNCTIONALITY

Introduction

The entire open labs idea is stemmed from the concept of open source technologies. The open labs hardware for version one was based on the Arduino since it is the most popular open source hardware development platform.

Version one of the open labs only consists of three components, an instrument board, client software and web application. The instrument hardware is where the signals are connected and the client PC is where software runs to send data to the server where the web application runs.

Instrument Module

The instrument hardware consists of an Arduino microcontroller and a signal conditioning circuit. The signal conditioning circuit is a simple resistor network that offsets the input signal voltage to 2.5V. This is to allow both negative and positive voltages to be read by the Analog to digital converter of the Arduino.

Signal conditioning circuit

The signal conditioning circuit as mentioned above simply offsets the zero line of the input waveform from 0V to 2.5V. An image of the signal conditioning circuit is shown in the image that follows.

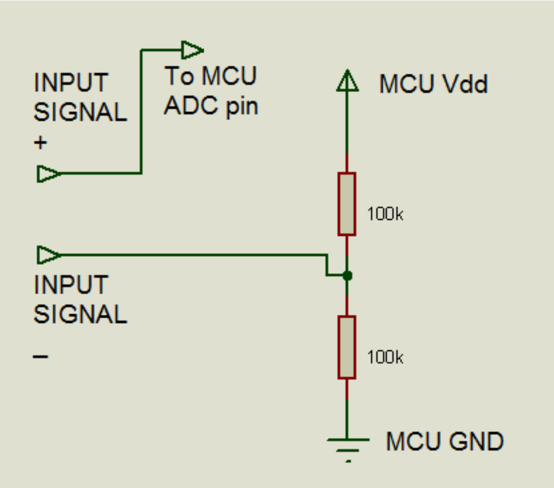


Figure 1: Signal conditioning circuit

The image also shows how the signal conditioning circuit connects to the Arduino.

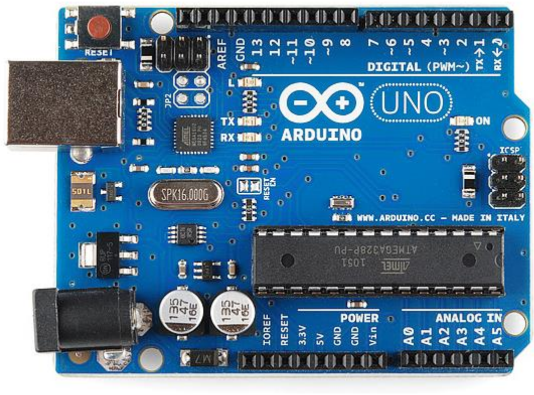
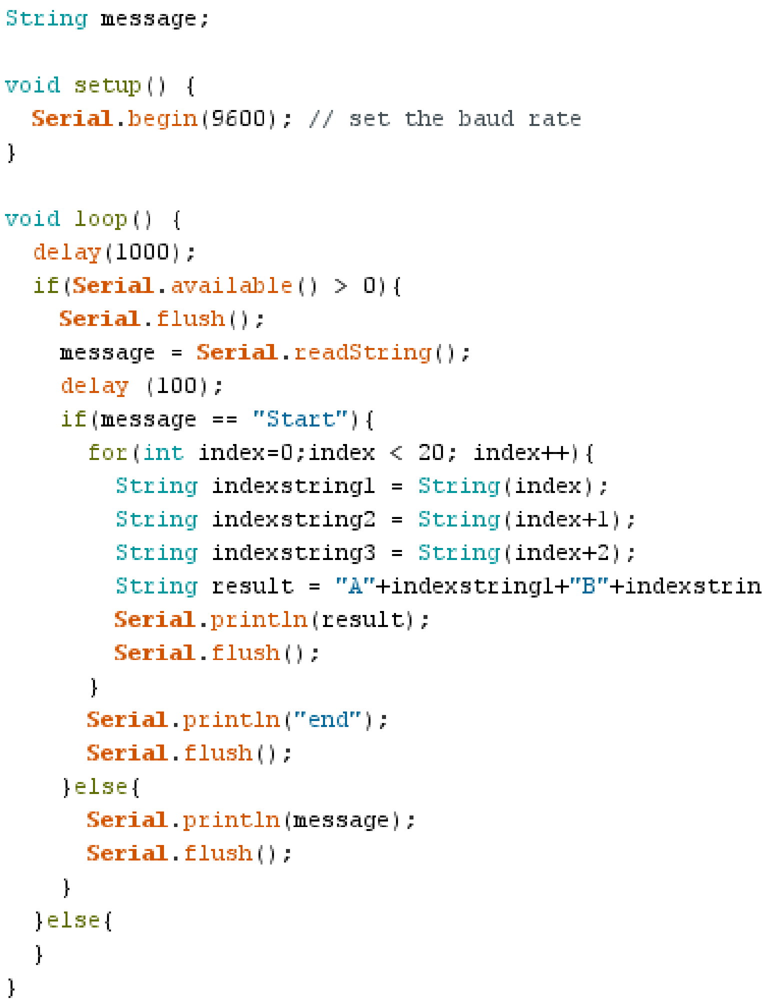


Figure 2: Arduino Uno

The arduino was programmed to send the analogue to digital conversion values via serial to the PC at a baud rate of 9600 repeatedly after receiving the trigger string “Start”. The following image shows the arduino code.



```
String message;

void setup() {
  Serial.begin(9600); // set the baud rate
}

void loop() {
  delay(1000);
  if(Serial.available() > 0){
    Serial.flush();
    message = Serial.readString();
    delay (100);
    if(message == "Start"){
      for(int index=0;index < 20; index++){
        String indexstring1 = String(index);
        String indexstring2 = String(index+1);
        String indexstring3 = String(index+2);
        String result = "A"+indexstring1+"B"+indexstrin
        Serial.println(result);
        Serial.flush();
      }
      Serial.println("end");
      Serial.flush();
    }else{
      Serial.println(message);
      Serial.flush();
    }
  }else{
  }
}
```

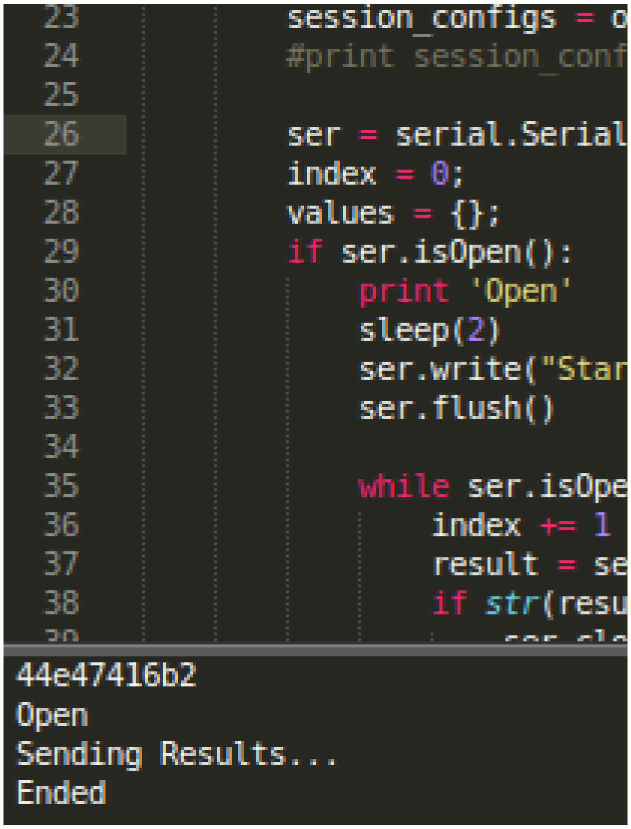
Figure 3: Arduino code

Client software

The Client software is basically an application that runs on a PC to which the instrument module is connected. It was developed in python in an UBUNTU environment so it only works on a PC running a Linux operating system. The PC on which the client software runs is called the client PC.

The client software which is just a python script acquires measured values from the instrument module through the USB port. These values start coming in immediately after the python script sends the trigger string. The values are in form of strings concatenated with individual values separated by letters A,B,C&D.

After receiving these values, the python script sends these values to the web application by HTTP posting every single time a set of values arrives. Thus, the client software cannot work without the client PC being connected to the internet.



```
        session_configs = o
        #print session_conf

        ser = serial.Serial
        index = 0;
        values = {};
        if ser.isOpen():
            print 'Open'
            sleep(2)
            ser.write("Star
            ser.flush()

            while ser.isOpe
                index += 1
                result = se
                if str(resu
```

```
44e47416b2
Open
Sending Results...
Ended
```

Figure 4: snippet of python script

Web Application

The web application is based upon several platforms which include PHP, MySQL, JavaScript, CSS and HTML.

PHP was used as the core server side scripting language. It plays the biggest role as it is the backend engine of the entire web application Therefore, it is responsible for manipulating the data that comes in from the PC.

The JavaScript, alongside CSS and HTML together constitute the web application’s user interface.

MySQL is an open source database with good performance, reliability and ease of use. The open labs web application uses MySQL for storing user credentials, user sessions and experiment configurations.

IV. LIMITATIONS

1) The hardware design of the instrument board of open labs version 1 cannot measure voltages above 2.5V or below -2.5V.
2) The hardware cannot sample input signals beyond a sampling rate of 125 kHz.
3) Version one of open labs only permits one way flow of data i.e. data from the instrument board can be viewed by a user over the internet but the user cannot make any adjustments or send data back to the lab server PC.
4) The data from the lab server is displayed in its raw format i.e. analog to digital conversion values instead of being graphed.
5) Version one of the open labs only facilitates one user at a time.

V. PERFORMANCE EVALUATION

The open labs version one can therefore be regarded as merely a proof of the concept of online laboratories based upon open source technologies. It cannot be used for actual engineering labs simply because it does not meet the requirements of such implementation both on the hardware and software front.