REPORT 6: DAQ INTERFACING WITH MICROCONTROLLERS.

GROUP 4

MCTA 3203

SEMESTER 2 2023/2024

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION:
18 MAY 2024

	NAME	MATRIC NUMBER
1.	NUR SHAHEERA BINTI KAMIS	2214040
2.	NORAMIRA HUSNA BT NORKHAIRANI	2214496
3.	MUHAMMAD EIQBAL BIN HASBOLLAH	2216911

TABLE OF CONTENT

ABSTRACT

This experiment investigates the usage of Arduino as a simple data acquisition (DAQ) tool for collecting data from LDR and LM35 sensors. Analogue sensor signals were translated into digital data via circuit setup and Arduino IDE programming. The verification and uploading of the code to the Arduino board, followed by the logging of sensor output in the PLX-DAQ spreadsheet, enabled detailed data analysis. Detailed code comments and instructive Excel graphs gave information about sensor integration and data analysis methods. This hands-on experience emphasises the importance of Arduino-based solutions in a variety of sectors, promoting a better grasp of sensor technology and data processing approaches.

Keywords: Arduino, data acquisition, LM35, LDR, PLX-DAQ, sensor integration, data analysis.

MATERIAL & EQUIPMENT

- PLX-DAQ
- Arduino Board
- LDR
- LM35
- Jumper Wires
- Resistor
- Breadboard

EXPERIMENTAL SETUP

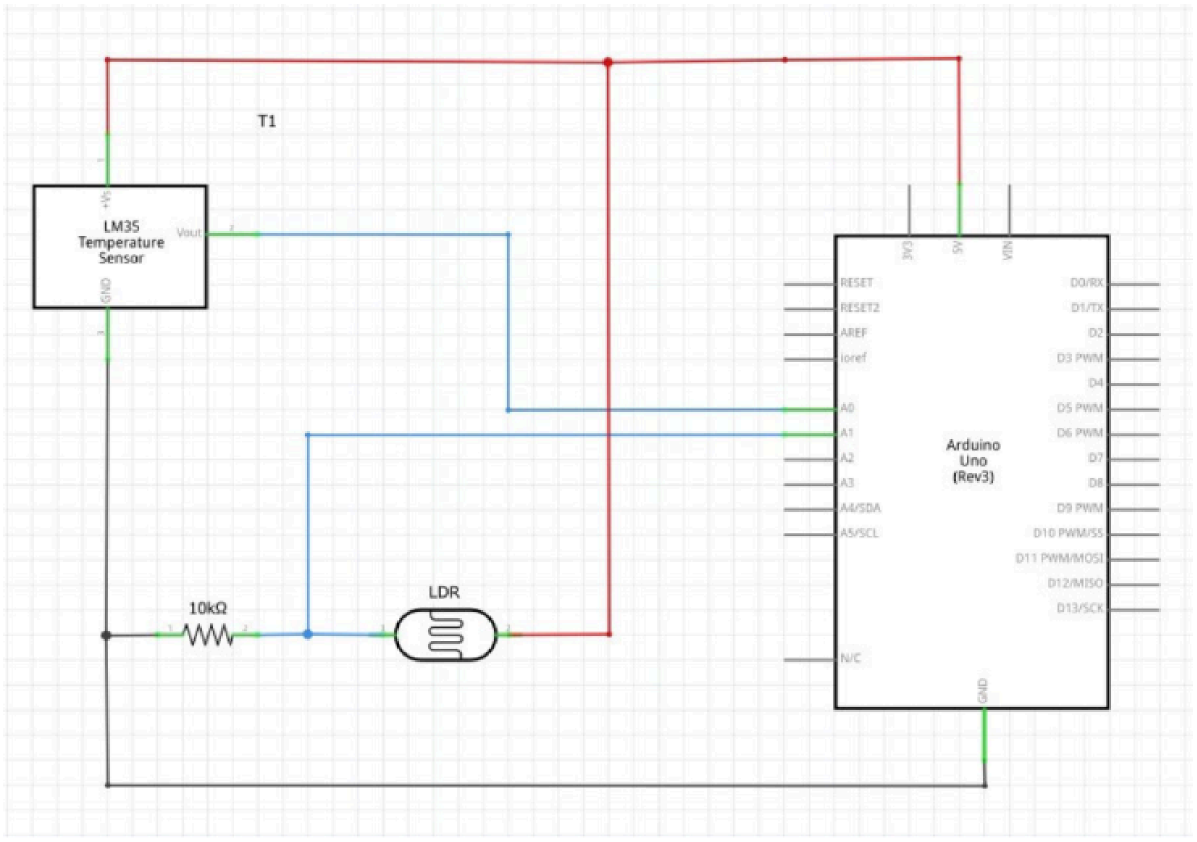


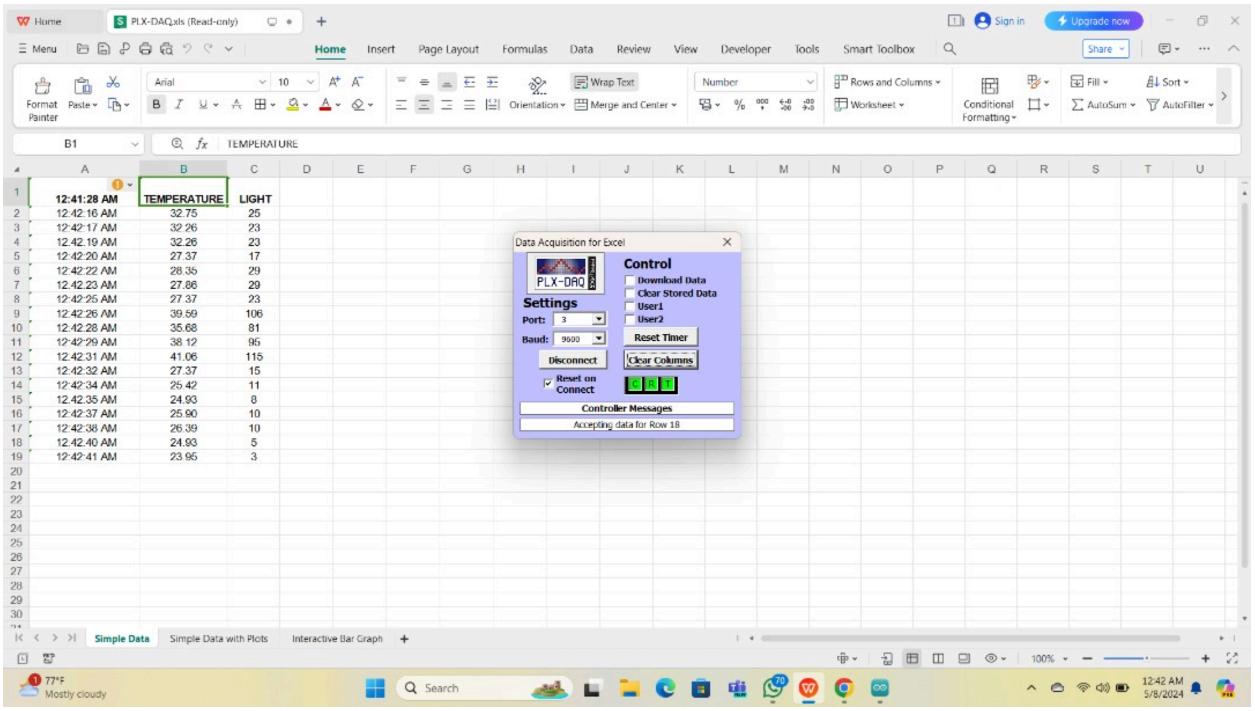
Fig. 3

METHODOLOGY/PROCEDURE

1. Code was created in Arduino IDE to read analogue signals from LM35 and LDR and convert them to digital.
2. The sample code provided below was completed. Alternatively, new code could have been created from the beginning.
3. The programmes were verified and uploaded to the Arduino board.
4. The PLX-DAQ spreadsheet was launched. The relevant COM port was selected, and sensor output was created in the spreadsheet.

5. Comments were used in the report to explain each line of code, and interesting Excel graphs were created using sensor data.

RESULTS



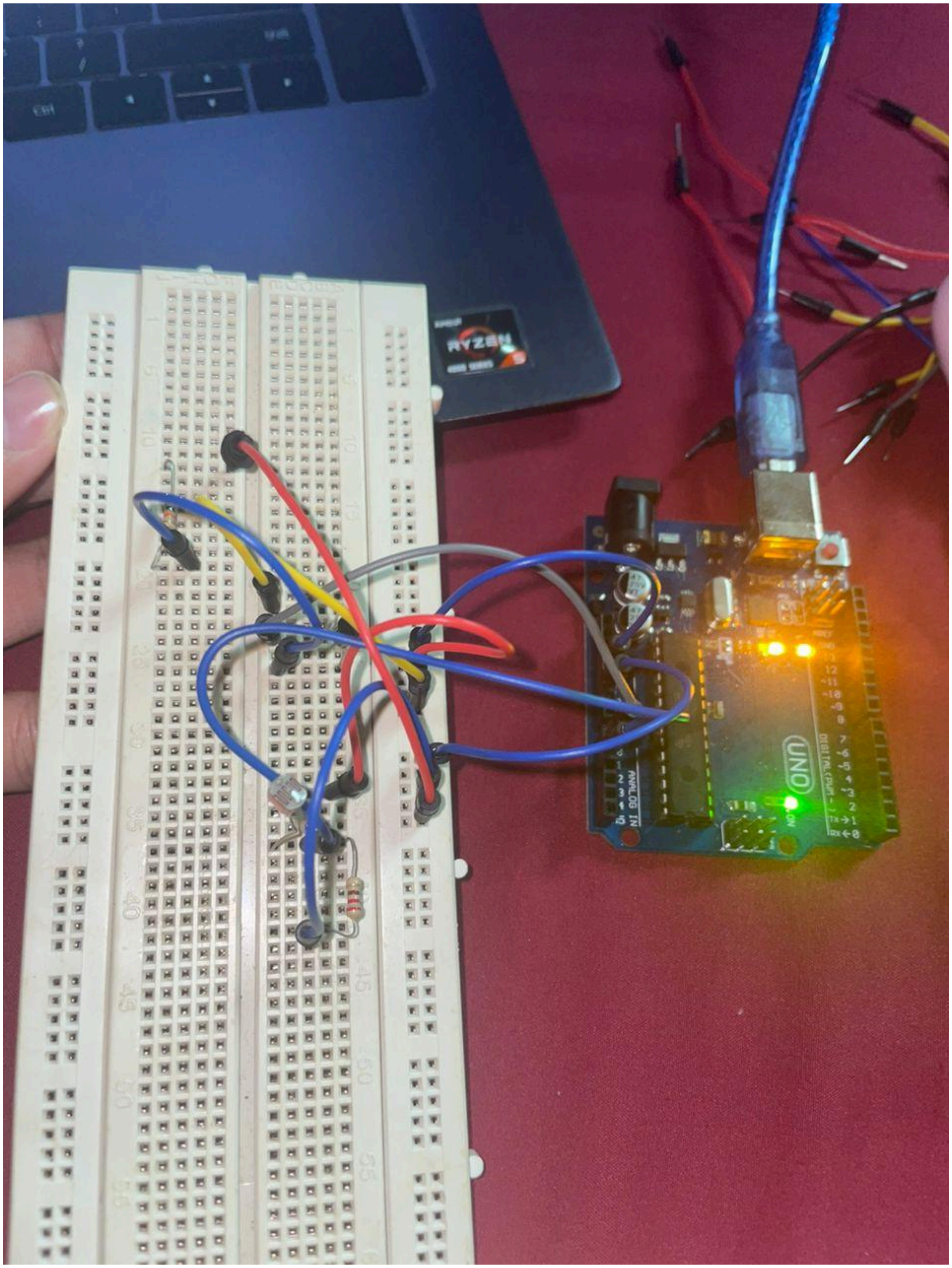
12:41:28 AM	TEMPERATURE	LIGHT
12:42:16 AM	32.75	25
12:42:17 AM	32.26	23
12:42:19 AM	32.26	23
12:42:20 AM	27.37	17
12:42:22 AM	28.35	29
12:42:23 AM	27.86	29
12:42:25 AM	27.37	23
12:42:26 AM	39.59	106
12:42:28 AM	35.68	81
12:42:29 AM	38.12	95
12:42:31 AM	41.06	115
12:42:32 AM	27.37	15
12:42:34 AM	25.42	11
12:42:35 AM	24.93	8
12:42:37 AM	25.90	10
12:42:38 AM	26.39	10
12:42:40 AM	24.93	5
12:42:41 AM	23.95	3

Data Acquisition for Excel

Settings
Port: 3
Baud: 9600
[✓] Reset on Connect

Control
[ ] Download Data
[ ] Clear Stored Data
[ ] User1
[ ] User2

Controller Messages
Accepting data for Row 18



DISCUSSION

- Hardware

Components of A Computer System

The experiment entails utilising Arduino as a data gathering instrument, establishing a connection with an LDR (Light Dependent Resistor) and an LM35 temperature sensor. Now, let's analyse the hardware components and their respective functions:

The Arduino board functions as the central component of the data acquisition system. The system receives analogue signals from the sensors, turns them into digital data, and transmits this data to the computer for additional analysis. Arduino Uno, Mega, or Nano are often utilised boards for such applications.

A Light Dependent Resistor (LDR) is a type of resistor that changes its resistance in response to the amount of light it receives. The resistance of a material reduces in strong light conditions, but it rises in darkness. This characteristic enables the Light Dependent Resistor (LDR) to function as a light sensor. The LDR is linked to one of the analog input pins on the Arduino, allowing it to read the fluctuating voltage corresponding to different light intensities.

The LM35 is a very accurate temperature sensor that produces an output voltage that is directly proportional to the temperature in Celsius. It gives a simple technique to measure temperature with an accuracy of ±0.5°C. The sensor outputs an analog voltage that the Arduino reads using another analog input pin.

Connecting Wires and Breadboard: These are used to construct the appropriate circuits to link the sensors to the Arduino. The breadboard enables for quick and flexible connections without the need for soldering.

Circuit Setup

Power Supply: Connect the Arduino to a power supply, often by USB from a computer, which also enables for data transfer and programming.

LDR Circuit: One end of the LDR is attached to a 5V supply on the Arduino.

The opposite end is linked to both an analog input pin (e.g., A0) and a resistor.

The opposite end of the resistor is linked to the ground (GND). This generates a voltage divider circuit where the voltage at the analog input pin fluctuates with the light intensity.

LM35 Circuit:

The LM35 has three pins: VCC, GND, and VOUT.

Connect VCC to the 5V supply on the Arduino.

Connect GND to the ground on the Arduino.

Connect VOUT to an analog input pin (e.g., A1). This pin will read the analog voltage proportional to the temperature.

- Electrical

Arduino Board: Acts as the central processing unit, delivering electricity and interpreting sensor data.

LDR (Light Dependent Resistor):

Power: Connect one terminal to the 5V pin of the Arduino.

Voltage Divider: Connect the other terminal to both an analog input pin (e.g., A0) and a resistor, with the other end of the resistor connected to ground (GND). This arrangement allows the Arduino to read different voltages dependent on light intensity.

LM35 Temperature Sensor:

VCC: Connect to the 5V pin on the Arduino.

GND: Connect to one of the ground (GND) pins.

VOUT: Connect to another analog input pin (e.g., A1) to receive the temperature data as an analog voltage.

<u>**Circuit Explanation**</u>

Voltage Divider for LDR: The combination of the LDR and a fixed resistor provides a voltage divider, where the voltage at the junction fluctuates with light intensity. This variable voltage is read by the analog pin A0 on the Arduino.

Analog Signal Conversion: The LM35 gives a linear analog output exactly proportional to the temperature, which the Arduino reads via analog pin A1.

Data Transmission

Serial Communication: The Arduino employs serial communication to deliver the digitized sensor data to a computer for logging and analysis, often utilising the USB connection.

- Software

<u>Arduino IDE Programming</u>

Setup and Initialization: Include Libraries: Use essential libraries for serial communication and sensor interfacing.

Pin Configuration: Define the analog input pins attached to the LDR and LM35 (e.g., const int LDRPin = A0; const int tempPin = A1;).

Serial Begin: Initialize serial communication in the setup() method (e.g., Serial.begin(9600);).

Reading Sensor Data:

Analog Read: Use the analogRead() function to read values from the analog pins (e.g., int lightValue = analogRead(LDRPin); int tempValue = analogRead(tempPin);).

Data Conversion:

LDR Data: Convert the analog value to a comparable light intensity (if needed).

LM35 Data: Convert the analog reading to temperature in Celsius (e.g., float temperature = (tempValue * 5.0 * 100.0) / 1024;).

Data Transmission:

Serial Print: Send the sensor data to the computer via the serial port (e.g., Serial.print("Light: "); Serial.print(lightValue); Serial.print(", Temp: "); Serial.println(temperature);).

PLX-DAQ Integration

Setup PLX-DAQ:

Connect Arduino: Ensure the Arduino is linked to the PC via USB.

Start PLX-DAQ: Open the PLX-DAQ spreadsheet and set it to read data from the relevant COM port.

Data Logging:

Real-Time Logging: PLX-DAQ logs the incoming data into an Excel spreadsheet, delivering real-time updates.

Data Formatting: Ensure the data format in the Arduino code matches the anticipated format in PLX-DAQ for accurate processing (e.g., comma-separated values).

Data Analysis

Graph Creation:

Excel Graphs: Use Excel’s graphing features to construct visual representations of the light intensity and temperature data over time.

<u>Commenting and Documentation:</u>

Code Comments: Provide descriptive comments in the Arduino code to clarify each step and function.

Excel Annotations: Use annotations in Excel to highlight key data points or patterns.

QUESTION

In your report, write the comment to explain each line of the codes and produce meaningful excel plots from the sensor’s data

```
float tempcelc;      // Variable to store temperature in Celsius
int ldr_value;       // Variable to store raw LDR value from analog input
int lm_value;        // Variable to store raw temperature sensor value from analog input


void setup() {
  Serial.begin(9600);              // Initialize serial communication at 9600 baud rate
  Serial.println("CLEARDATA");       // Clear any existing data in the PLX-DAQ spreadsheet
  Serial.println("LABEL, TIME, TEMPERATURE, LIGHT"); // Set column labels in the
PLX-DAQ spreadsheet
}
void loop() {
  lm_value = analogRead(A0);        // Read the analog value from the LM35 temperature sensor
connected to A0
```

```
  tempcelc = (lm_value / 1023.0) * 5000.0; // Convert the raw value to millivolts
  tempcelc = tempcelc / 10.0;        // Convert millivolts to degrees Celsius (10mV per degree
Celsius)
  ldr_value = analogRead(A1);        // Read the analog value from the LDR connected to A1
  Serial.print("DATA, TIME, ");      // Send the prefix for PLX-DAQ to recognize data input
  Serial.print(tempcelc);            // Send the temperature value
  Serial.print(",");                 // Separator between temperature and light values
  Serial.println(ldr_value);         // Send the LDR value and end the line
  delay(1500);                       // Wait for 1.5 seconds before repeating the loop
}
```

1. Setting Up PLX-DAQ

- Ensure PLX-DAQ is installed and configured correctly to receive data from the correct COM port.
- Open the PLX-DAQ Excel template and start the data logging.

2. Data Logging

- The data will be automatically logged into the Excel spreadsheet with columns labeled as TIME, TEMPERATURE, and LIGHT.
- Each row will correspond to a data entry with the current time, the temperature in Celsius, and the raw LDR value.

3. Creating Excel Plots

- **Temperature Plot**:
    1. Select the data range for TIME and TEMPERATURE columns.
    2. Insert a line chart to visualize temperature variations over time.
    3. Label the axes (e.g., X-axis: Time, Y-axis: Temperature (°C)) and add a title to the chart.
- **Light Intensity Plot**:
    1. Select the data range for TIME and LIGHT columns.
    2. Insert a line chart to visualize light intensity variations over time.
    3. Label the axes (e.g., X-axis: Time, Y-axis: Light Intensity (Raw Value)) and add a title to the chart.

4. Analyzing the Plots

- **Temperature Plot Analysis**:
    - Look for trends and patterns in temperature changes over time.
    - Identify any significant spikes or drops that may indicate environmental changes.
- **Light Intensity Plot Analysis**:
    - Observe the variations in light intensity captured by the LDR.
    - Correlate light intensity changes with potential external factors (e.g., changes in ambient lighting).

CONCLUSION

In conclusion, this experiment illustrated the use of Arduino as a basic data acquisition (DAQ) tool for collecting data from LDR and LM35 sensors. The analogue signals from the sensors were processed and transformed into digital data using circuit setup and Arduino IDE programming. Verification, uploading, and subsequent tracking of sensor output in the PLX-DAQ spreadsheet allowed for complete data analysis. Detailed code comments and informative Excel graphs gave useful information about sensor integration and data analysis approaches. This hands-on experience emphasises the importance of Arduino-based solutions in a variety of sectors, promoting a better grasp of sensor technology and data processing approaches.

RECOMMENDATION

To improve this experiment, it may be useful to include more sensors or broaden the range of data obtained. This might lead to a better knowledge of Arduino's data-collecting and analysis capabilities. Incorporating real-world applications or scenarios into the experiment may also provide students with a better understanding of how these talents are used in practical contexts. Furthermore, offering instruction on how to handle typical errors seen during circuit design or programming might assist students acquire the problem-solving abilities required in electronics and programming. Finally, conducting a conversation or reflection session following the experiment may encourage students to share their findings and perspectives, promoting a collaborative learning environment.

ACKNOWLEDGEMENT

A special thanks goes out to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

STUDENT DECLARATION

<u>Certificate of Originality and Authenticity</u>

This is to certify that we are responsible for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us.**

Signature: [illegible]	**Read**	/
Name: Noramira Husna Bt Norkhairani	**Understand**	/
Matric No: 2214496	**Agree**	/

Signature: [illegible]	**Read**	/
Name: Nur Shaheera Bt Kamis	**Understand**	/
Matric No:2214040	**Agree**	/

Signature: [illegible]	**Read**	/
Name: Muhammad Eiqbal bin Hasbollah	**Understand**	/
Matric No: 2216911	**Agree**	/