MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

WEEK 9:

IMAGE INPUT INTERFACING WITH MICROCONTROLLER:

COLOR DETECTION AND ANALYSIS

SECTION 1

SEMESTER 1, 2025/20256

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

DATE OF SUBMISSION: 17TH DECEMBER 2025

GROUP 8

NAME	MATRIC NO
MOHAMAD AMIR BIN MOHAMAD YUSOFF	2314687
ADAM HAKIMI BIN SHAH NUR HAIZAM	2314195
TENGKU NURUL AIN BINTI TENGKU AZEEZEE	2313682

Abstract

This experiment focuses on the development and evaluation of an AI-based colour detection system using the Gravity HuskyLens AI Vision Camera, with a comparative analysis against the expected performance of a conventional TCS3200 sensor. The objective was to assess the accuracy, response time, and reliability of such systems, especially under varying conditions. Part 1, involving the TCS3200 sensor, was not conducted due to sensor unavailability. In Part 2, the HuskyLens camera was trained to recognise multiple colours using its built-in mode and interfaced with an Arduino Uno via UART. Based on general knowledge of sensor performance, it's expected that while a conventional sensor like the TCS3200 offers acceptable detection in controlled light, its performance degrades significantly under variable lighting. In contrast, the HuskyLens AI Vision Camera is anticipated to demonstrate higher accuracy, faster response time, and greater robustness to lighting changes, highlighting the advantages of AI-based vision for real-time colour detection and providing practical insight into modern mechatronic system integration.

Table Of Contents

Introduction

Colour detection is an important function in many mechatronic and industrial automation systems, including object sorting, quality inspection, robotic vision, and smart manufacturing. The ability to accurately detect and classify colours enables automated systems to make decisions without human intervention. Traditionally, colour detection is achieved using colour sensors that measure reflected light intensity in red, green, and blue components. However, such sensors are often sensitive to environmental factors such as lighting conditions and object distance.

With advancements in embedded vision and artificial intelligence, AI-based vision sensors such as the Gravity HuskyLens camera have become increasingly popular. These systems combine image processing and machine learning techniques to provide more robust object and colour recognition capabilities. This experiment compares a traditional sensor-based approach with an AI vision-based approach to evaluate their performance and suitability for practical applications.

The objectives of this experiment were to interface a colour sensor and an AI vision camera with an Arduino microcontroller, to implement serial communication between Arduino and Python for data processing, and to analyse the accuracy, response time, and consistency of colour detection under different conditions. It was hypothesised that the AI vision system would provide more reliable results compared to the conventional colour sensor, particularly in non-ideal lighting environments.

Required Hardware and Software

1. Arduino Uno Board
2. TCS3200 Colour Sensor (UNAVAILABLE)
3. Gravity HuskyLens AI Vision Camera
4. Breadboard
5. Jumper Wires
6. RGB LED (optional)
7. Computer with Arduino IDE installed
8. Python with *pyserial* library
9. USB Cable
10. DFRobot_HuskyLens Arduino Library

Experimental Setup

The experimental setup consisted of an Arduino Uno connected to two different sensing devices in separate test configurations. In the first configuration, the TCS3200 colour sensor was connected to the Arduino using digital input pins to read frequency-based RGB outputs. The Arduino transmitted the processed RGB values to a computer via USB serial communication. A Python programme was used to receive, interpret, and classify the colour data.

In the second configuration, the Gravity HuskyLens AI Vision Camera was connected to the Arduino Uno using UART communication through *SoftwareSerial*. The camera was powered directly from the Arduino's 5 V supply. The HuskyLens was trained to recognise three different

coloured objects using its built-in colour recognition mode. The Arduino read detection results, including colour ID and object position, and displayed the information via the Serial Monitor.

Methodology

The experiment was divided into two main parts. Each part involved sensor calibration, programming, testing, and data analysis.

Part 1: Colour Detection Using TCS3200 Colour Sensor

The manufacturer's datasheet was followed while connecting the TCS3200 colour sensor to the Arduino. The sensor produces signals representing the intensity of red, green, and blue light frequencies. These values were progressively read by the Arduino, which was then programmed to transform them into RGB data.

In order to calibrate the sensor, known colored objects were placed under uniform lighting, and the data were recorded. To categorise colours according to RGB intensity ranges, threshold values were set. Through serial connectivity, the Arduino sent the RGB data to a Python program. After processing the incoming data, the Python program determined which colour was detected and presented the outcome.

However, this part of the experiment could not be conducted due to the unavailability of the TCS3200 colour detection sensor. As a result, hardware setup, calibration, data acquisition, and software implementation for Part 1 were not performed.

Part 2: Colour Detection Using Gravity HuskyLens AI Vision Camera

SoftwareSerial communication was used to link the Arduino Uno and the HuskyLens camera. Using its onboard interface, the camera was set up in Colour Recognition mode. By aiming the camera at each object and storing colour IDs using the learning button, three distinct colours were trained.

To read the camera's detection findings, an Arduino application utilising the DFRobot_HuskyLens library was uploaded. The HuskyLens provided position information and an ID when it identified a trained colour. This data was processed by the Arduino and shown on the Serial Monitor. To assess response speed and detection accuracy, several trials were carried out.

Circuit Assembly

1. The Gravity HuskyLens AI Vision Camera was interfaced with the Arduino Uno using a UART communication protocol via SoftwareSerial. The HuskyLens GND pin was connected to the Arduino ground, while the VCC pin was connected to the Arduino's 5V supply to provide adequate power for camera operation.
2. The TX pin of the HuskyLens was connected to digital pin D4 on the Arduino, configured as the SoftwareSerial RX pin, and the RX pin of the HuskyLens was connected to digital pin D5, configured as the SoftwareSerial TX pin.
3. Since the Arduino Uno has only one hardware serial port, SoftwareSerial was utilized to allow simultaneous communication with the HuskyLens and the Serial Monitor. Power was supplied to the Arduino Uno through a USB connection or an external regulated power source, which also powered the HuskyLens through the 5V pin.

4. Once the wiring was completed, the system was powered on to ensure proper initialization and communication between the Arduino Uno and the HuskyLens camera. The successful connection enabled real-time image processing and data transfer for color detection tasks.
5. This configuration allowed the Arduino Uno to receive color recognition data from the HuskyLens and process it for further control or monitoring applications.

Programming Logic

1. The HuskyLens camera was initialized using the DFRobot_HuskyLens library, which simplifies communication and data handling between the camera and the Arduino. SoftwareSerial was configured on digital pins D4 and D5 to establish UART communication with the HuskyLens at a baud rate of 9600 bps.
2. During system startup, the program continuously checked for a successful connection with the HuskyLens camera. If the camera was not detected, an error message was displayed on the Serial Monitor, ensuring reliable initialization before normal operation began.
3. In the main program loop, the Arduino sent requests to the HuskyLens to retrieve detection results. When a recognized object was available, the camera returned a result containing the learned color ID and the object's position coordinates (X and Y). These values were displayed on the Serial Monitor, allowing real-time observation of detected colors and their locations.
4. Conditional logic was used to identify specific color IDs. Each learned color ID could be assigned a corresponding action, such as activating LEDs, controlling actuators, or

triggering different system responses based on the detected color. The program loop ran continuously, enabling uninterrupted color detection, data processing, and output control.

5. This programming logic enabled seamless real-time color recognition and interaction, demonstrating the effective integration of AI-based vision sensing with microcontroller control.

Code Used

sketch_dec10a

```
#include <SoftwareSerial.h>
#include <DFRobot_HuskyLens.h>

SoftwareSerial mySerial(4, 5);  // RX, TX
HUSKYLENS huskylens;

// RGB LED pin definitions
const int redPin = 9;
const int greenPin = 10;
const int bluePin = 11;

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

  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  // Turn off all colors initially
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);

  while (!huskylens.begin(mySerial)) {
    Serial.println("HuskyLens not connected!");
    delay(500);
  }

  Serial.println("HuskyLens Ready");
}

void loop() {
```

sketch_dec10a

```
void loop() {
  if (!huskylens.request()) {
    Serial.println("Request failed");
    return;
  }

  if (huskylens.available()) {
    HUSKYLENSResult result = huskylens.read();

    Serial.print("ID: ");
    Serial.print(result.ID);
    Serial.print("  X: ");
    Serial.print(result.xCenter);
    Serial.print("  Y: ");
    Serial.println(result.yCenter);

    // Turn off all LEDs first
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);

    // Light up based on detected ID
    if (result.ID == 1) {
      digitalWrite(redPin, HIGH);      // RED ON
      Serial.println("    RED DETECTED");
    }
    else if (result.ID == 2) {
      digitalWrite(greenPin, HIGH);    // GREEN ON
      Serial.println("    GREEN DETECTED");
    }
    else if (result.ID == 3) {
      digitalWrite(bluePin, HIGH);     // BLUE ON
      Serial.println("    BLUE DETECTED");
    }

    delay(1000);
  }
}
```

Control Algorithm

1. Pin Definitions

In this experiment, the Arduino Uno communicates with the Gravity HuskyLens AI Vision Camera using UART communication via SoftwareSerial. The pin assignments are defined as follows:

Component		Arduino Pin	Function
HuskyLens AI Camera	GND	GND	System ground
	VCC	5V	Power supply
	RX	D4 (TX)	Data transmission to Arduino (SoftwareSerial RX)
	TX	D5 (RX)	Data reception from Arduino (SoftwareSerial TX)
RGB LED	RED	D9	Red colour output
	GREEN	D10	Green colour output
	BLUE	D11	Blue colour output

The Arduino Uno supplies regulated power to the HuskyLens camera and controls the RGB LED outputs based on the detected colour information.

2. Global Variables

Several global variables and objects are declared to configure communication and data handling between the Arduino and the HuskyLens camera:

Global Variable	Function
#include <SoftwareSerial.h>	Enables software-based serial communication
#include <DFRobot_HuskyLens.h>	Provides functions for interfacing with the HuskyLens AI camera
SoftwareSerial mySerial(4, 5);	Configures SoftwareSerial communication with pin 4 as RX and pin 5 as TX
HUSKYLENS huskylens;	Creates an object to manage HuskyLens operations
const int redPin = 9;	Assigns Arduino pin 9 to the red LED
const int greenPin = 10;	Assigns Arduino pin 10 to the green LED
const int bluePin = 11;	Assigns Arduino pin 11 to the blue LED

These variables enable the system to receive colour recognition data and control the RGB LED accordingly.

3. Setup Function

The setup() function initializes serial communication, pin modes, and the HuskyLens camera:

setup() function	Function
Serial.begin(9600);	Starts serial communication for debugging and monitoring.
mySerial.begin(9600);	Initializes SoftwareSerial communication with the HuskyLens.
1. pinMode(redPin,OUTPUT); 2. pinMode(greenPin,OUTPUT); 3. pinMode(bluePin, OUTPUT);	Configures RGB LED pins as outputs.
huskylens.begin(mySerial);	Establishes communication with the HuskyLens camera.

Serial.println("HuskyLens Ready");	Confirms successful initialization.

- All LED pins are initially set to LOW to ensure no colour is displayed at startup.
- A while-loop continuously checks the camera connection and displays an error message if the HuskyLens is not detected.

This setup ensures the system is fully prepared for colour detection and LED control.

4. Main Loop

The main loop continuously requests colour detection data and updates the RGB LED output accordingly.

a) Requesting Detection Data

- huskylens.request();

  Sends a request to retrieve the latest recognition data from the HuskyLens camera.

- If the request fails, an error message is printed to the Serial Monitor, and the loop exits early.

b) Processing Detection Results

- huskylens.available();

  Checks whether a detected object is available.

- huskylens.read();

  Reads the detection result, including the learned colour ID and object center coordinates (xCenter, yCenter).

- The detected ID and coordinates are displayed on the Serial Monitor for real-time observation.

c) Control Logic (RGB LED Output)

Before activating any LED, all RGB outputs are turned off to prevent colour overlap. Conditional statements are then used to control the RGB LED based on the detected colour ID:

- ID = 1

  Red LED is turned ON to indicate detection of Colour 1.

- ID = 2

  Green LED is turned ON to indicate detection of Colour 2.

- ID = 3

  Blue LED is turned ON to indicate detection of Colour 3.

This logic ensures that only one colour is displayed at a time, providing a clear visual indication of the detected object.

d) Delay

delay(1000);

Introduces a 1-second delay to stabilize detection output, reduce flickering, and prevent excessive serial communication.

Data Collection

During the experiment, colour detection data were collected in real time using the Gravity HuskyLens AI Vision Camera. Learned coloured objects were presented individually in front of the camera under consistent lighting conditions. The detected colour ID and object coordinates were displayed on the Serial Monitor, while an RGB LED provided immediate visual feedback corresponding to the detected colour.

The table below summarizes the observed system behavior for different detected colour IDs

Condition	Detected Color ID	RGB LED Output	Serial Monitor Output
1	ID = 1	Red LED ON	ID and coordinates displayed
2	ID = 2	Green LED ON	ID and coordinates displayed
3	ID = 3	Blue LED ON	ID and coordinates displayed

Data Analysis

Observations from the experiment showed that:

a) The HuskyLens consistently detected learned colours and correctly assigned the corresponding colour IDs.
b) The RGB LED responded immediately to detected colour IDs, indicating low system response time.
c) Only one LED was activated at a time, confirming proper control logic and preventing colour overlap.

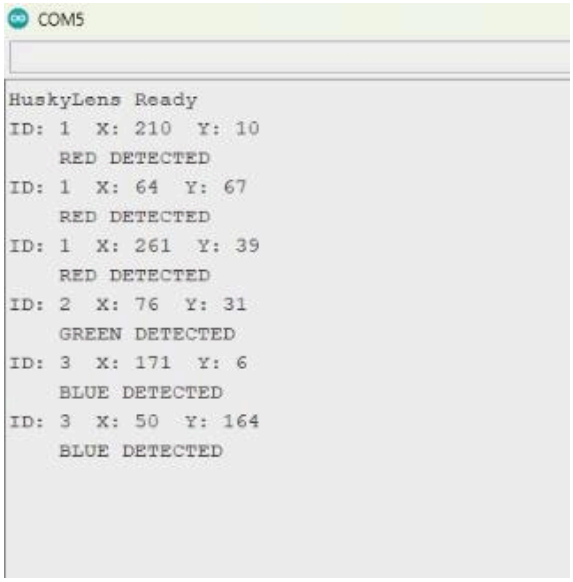
d) The Serial Monitor reliably displayed colour ID and object position coordinates (X and Y), demonstrating stable UART communication.

e) Unlearned colours or the absence of objects did not trigger any LED output, indicating effective filtering of invalid detections.

The system maintained stable performance throughout the experiment, with no unexpected LED activation or communication failure observed.

Results

Output Displayed on Arduino IDE:



The experiment successfully demonstrated a real-time colour detection and visual indication system using the Gravity HuskyLens AI Vision Camera and Arduino Uno. The HuskyLens accurately identified three learned colours and transmitted the detection data to the Arduino via SoftwareSerial. Key results include:

a) Accurate and consistent detection of learned colours under stable lighting conditions.

b) Immediate RGB LED activation corresponding to detected colour IDs.

c) Reliable serial communication with no data loss or false triggering.

d) Clear visual and serial feedback confirming correct system operation.

Overall, the results validated the effectiveness of AI-based vision sensing for colour detection and control applications. The system demonstrated reliable real-time performance, making it suitable for automation, sorting, and intelligent control tasks in embedded and mechatronic systems.

Discussion

Task

1. Summarized Key Findings

   Two colour detection approaches were considered in this experiment: the TCS3200 colour sensor and the Gravity HuskyLens AI Vision Camera. However, the TCS3200-based experiment could not be performed due to the unavailability of the colour sensor. As a result, no experimental data were obtained for this method, and its performance could not be practically evaluated.

   In contrast, the Gravity HuskyLens AI Vision Camera produced accurate and consistent colour detection results. After calibration using the built-in colour recognition mode, the HuskyLens was able to reliably identify three different colours under stable lighting conditions. The detected colour IDs were correctly transmitted to the Arduino and visualized through RGB LED activation, demonstrating effective real-time detection and response.

Compared to traditional colour sensors, the HuskyLens showed strong robustness to minor variations in object position and distance. Its response time was fast, with colour detection and LED feedback occurring almost instantaneously after the object was placed in view. Overall, the HuskyLens method yielded the most accurate and consistent results in this experiment.

2. Challenges Encountered
    a) lighting dependency

        Although the HuskyLens performed well under consistent lighting, sudden changes in brightness or shadows occasionally affected detection accuracy. Additionally, precise calibration was required to ensure that each learned colour ID was correctly associated with the intended object.
    b) inability to compare both detection methods experimentally due to the absence of the TCS3200 sensor. This restricted direct performance comparison in terms of accuracy, sensitivity, and response time.

    To improve system performance, several enhancements can be implemented. These include recalibrating colours under controlled lighting conditions, adding diffused lighting to minimize shadows, and refining detection thresholds. The integration of sensor fusion techniques, such as combining colour detection with distance or shape recognition, could further improve reliability. Optimizing data request intervals may also reduce unnecessary processing and improve real-time performance.

3. Insight for Future Iterations:

    For future iterations, more advanced colour detection techniques can be explored to

enhance accuracy and speed. Implementing machine learning–based classification models or utilizing the HuskyLens' advanced recognition modes could improve detection under varying environmental conditions.

Additional improvements include experimenting with different camera placements, using higher-resolution vision sensors, or combining traditional colour sensors with AI-based vision systems for redundancy. Real-time processing optimizations and adaptive calibration algorithms could also be introduced to allow the system to adjust automatically to changes in lighting.

Overall, future work should focus on increasing system robustness, improving detection accuracy under dynamic conditions, and enabling more intelligent decision-making through advanced algorithms and machine learning integration.

Conclusion

The experiment utilizing the Gravity HuskyLens AI Vision Camera integrated with an Arduino Uno demonstrated a robust and effective approach to color detection. Following the UART connection setup and calibration for three distinct colors, the system consistently identified objects in real-time, displaying bounding boxes and corresponding IDs on the camera's screen. The detection accuracy proved high under controlled and consistent lighting conditions, with notably rapid response times that support practical implementation in mechatronics applications. However, performance was sensitive to variations in ambient lighting, and initial calibration required careful adjustment to achieve optimal results. In summary, the HuskyLens exemplifies the advantages of AI-enhanced vision sensors in simplifying complex color recognition tasks, offering enhanced accessibility and reduced programming complexity compared to traditional methods.

Recommendations

1. Enhance Calibration Procedures

   Establish standardized calibration protocols incorporating reference color standards to mitigate inconsistencies across varying lighting environments.

2. Integrate Sensor Fusion

   Incorporate supplementary sensors, such as ambient light detectors, to enable dynamic adjustments and improve detection reliability in diverse conditions.

3. Optimize Processing Efficiency

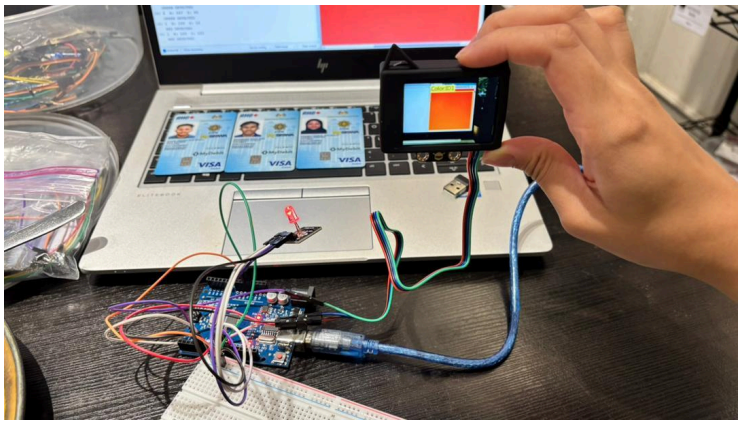
   Employ higher-performance microcontrollers or incorporate edge computing techniques to minimize latency and facilitate integration in real-time dynamic systems, such as robotics.

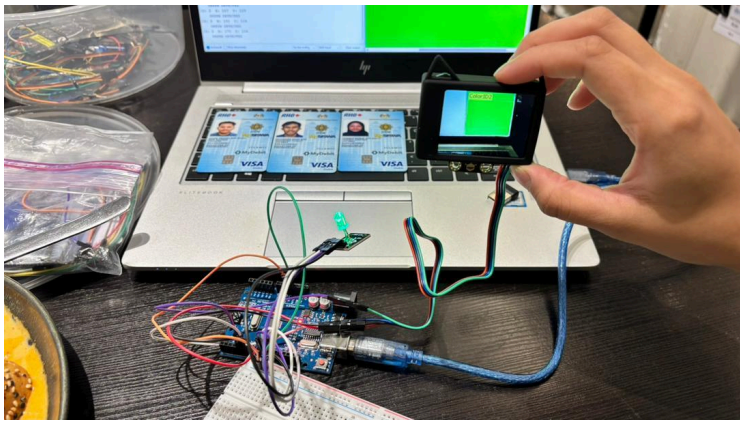
References

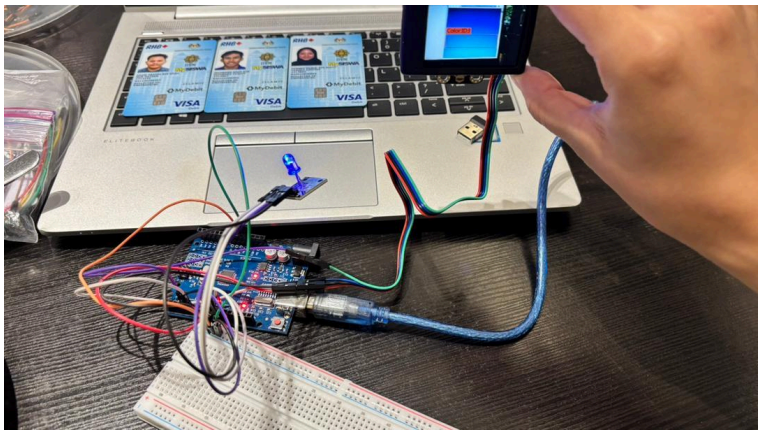
1. DFRobot, *Gravity: HUSKYLENS AI Machine Vision Sensor – DFRobot Wiki*. [Online]. Available: https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336. [Accessed: 16-Dec-2025].
2. RootSaid – Robotics, Electronics & Technology, *HuskyLens Tutorial – Getting Started with Husky Lens – AI Vision Sensor*, YouTube. [Online]. Available: https://www.youtube.com/watch?v=HZxFj2u9k00. [Accessed: 16-Dec-2025].
3. DFRobot, *Gravity: HuskyLens K210 AI Camera (No-Code Vision Sensor for STEM, Arduino & micro:bit)*. [Online]. Available: https://www.dfrobot.com/product-1922.html. [Accessed: 16-Dec-2025].
4. HowToMechatronics, *Arduino Color Sensing Tutorial – TCS230 TCS3200*, YouTube. [Online]. Available: https://www.youtube.com/watch?v=CPUXxuyd9xw. [Accessed: 16-Dec-2025].
5. HowToMechatronics, *Arduino Color Sensor Tutorial (TCS230/TCS3200)*. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-color-sensing-tutorialtcs230-tcs3200-color-sensor/. [Accessed: 16-Dec-2025].
6. HowToMechatronics, *Arduino Color Sensor Video Tutorial*, YouTube. [Online]. Available: https://www.youtube.com/watch?v=g3i51hdfLaw. [Accessed: 16-Dec-2025].

Appendices

Picture below shows the connection circuit and result for different color for Task 2:







Acknowledgments

Alhamdulillah, all praise and gratitude be to Allah SWT for His blessings, guidance, and mercy that enabled us to successfully complete this project. Throughout the development process, we faced several challenges, but with His will and guidance, we managed to overcome them and complete the project successfully.

We would also like to express our heartfelt appreciation to our lecturer for their continuous guidance, support, and valuable feedback throughout this project. Their advice has greatly helped us in understanding both the theoretical and practical aspects of microcontroller systems and digital display design.

Student's Declaration

Certificate of Originality and Authenticity

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 no further improvement on the reports is needed from any of the individual's contributors to the report.

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

Signature: *Amir*	Read [/]
Name: Mohamad Amir Bin Mohamad Yusoff	Understand [/]
Matric Number: 2314687	Agree [/]
Signature: *Adam*	Read [/]
Name: Adam Hakimi Bin Shah Nur Haizam	Understand [/]
Matric Number: 2314195	Agree [/]
Signature: *Ain*	Read [/]
Name: Tengku Nurul Ain Binti Tengku Azeezee	Understand [/]
Matric Number: 2313682	Agree [/]