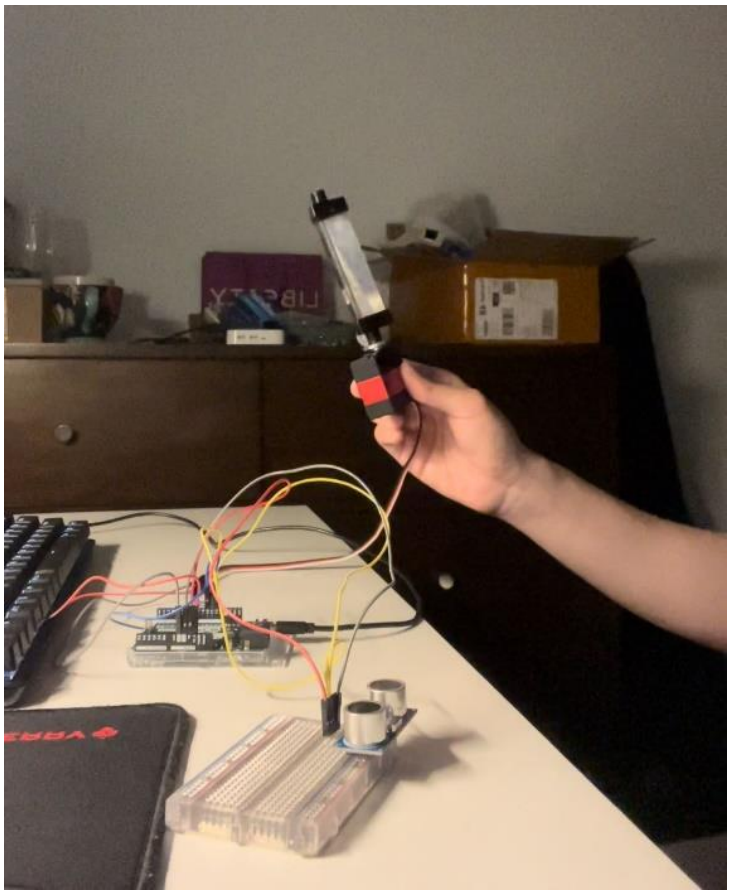
9.14-9.21

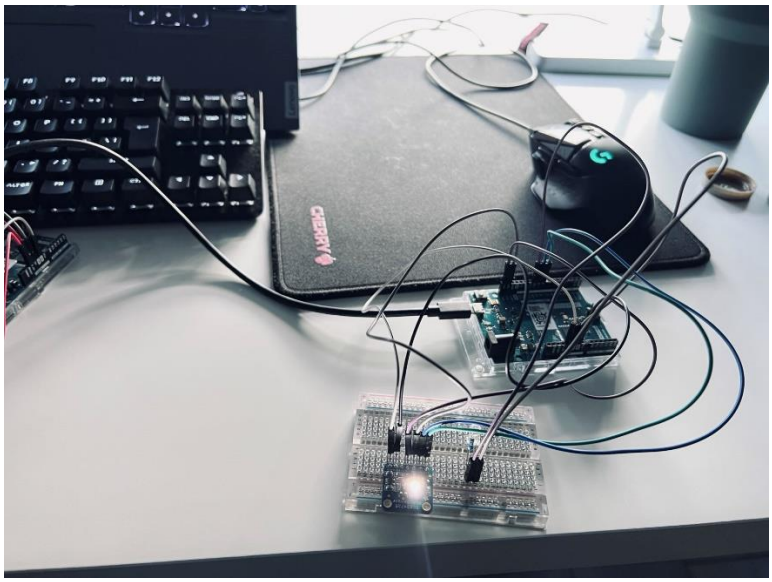
At this stage I continued to work on the preparation of the device. When considering how to receive the light refracted from the prism, I initially considered using a photoresistor, which I had covered in class. However, based on my experience in previous projects, I realised that the photoresistor had a very limited range of light reception and the light source would need to be almost close to the resistor to trigger it. To solve this problem, I turned to Lexin, our school's technical support teacher, who recommended the RGB colour sensor, and I borrowed a TFT screen for video playback.

In subsequent tests, I encountered a new challenge. I discovered that when connected to a power source, the RGB colour sensor (TSC 34725) has a built-in light source that was originally intended to illuminate objects to receive colour feedback. However, when I used this sensor as a photosensitive element, the built-in LED greatly interfered with data reception and I was faced with the dilemma of not being able to turn off this LED. Meanwhile, I started testing TFT screens. In my initial attempts, I managed to start the screen and draw patterns by searching the web for open source code, but I ran into problems when trying to import pictures or videos into the TFT screen. Neither importing the image via the Arduino nor converting the image to hexadecimal format was successful. I then tried importing using an SD card, but the screen was never able to read the contents of the SD card.

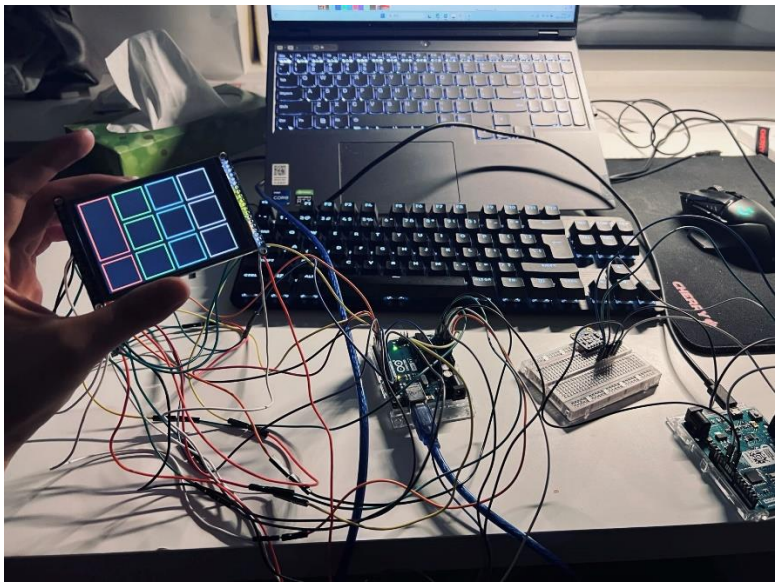
In an attempt to resolve these issues, Matt and I arranged a tutorial session, but unfortunately the problem remained unsolved. As a result, I began to consider switching to a Raspberry Pi to control the RGB colour sensor and TFT screen sections.



Distance sensor & Servo & Prism



RGB color sensor & LED test



TFT screen test

Arduino Leonardo

DistanceSersor.ino

```ino
#include <Servo.h>

#define PIN_TRIG 12
#define PIN_ECHO 11
#define SERVO_PIN 10

Servo myservo;
float cm;
float temp;

void setup() {
  Serial.begin(9600);
  pinMode(PIN_TRIG, OUTPUT);
  pinMode(PIN_ECHO, INPUT);
  myservo.attach(SERVO_PIN);
  myservo.write(90); // 设置舵机初始角度为90度
}

void gradualMove(int currentPos, int newPos) {
  int stepDelay = 35; // 控制舵机移动速度的延迟时间，数值越大移动越慢
  if (currentPos < newPos) {
    for (int pos = currentPos; pos <= newPos; pos++) {
      myservo.write(pos);
      delay(stepDelay);
    }
  } else {
    for (int pos = currentPos; pos >= newPos; pos--) {
      myservo.write(pos);
      delay(stepDelay);
    }
  }
}

void loop() {
  digitalWrite(PIN_TRIG, LOW);
```

输出

行 13，列 29　Arduino Leonardo 在COM6上 [没有连接]

Arduino Leonardo

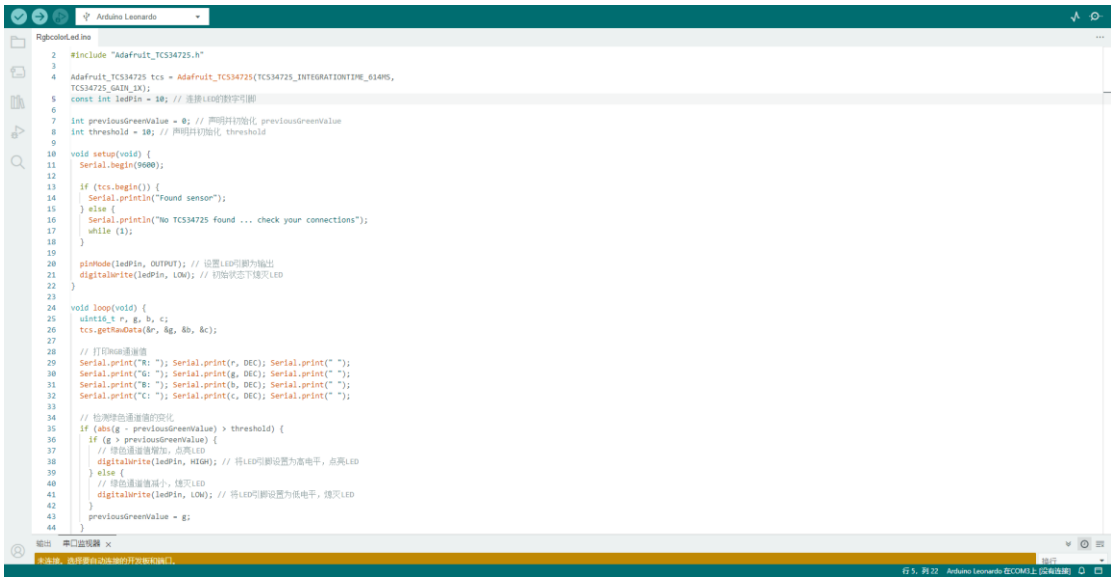
DistanceSersor.ino

```ino
    }
  } else {
    for (int pos = currentPos; pos >= newPos; pos--) {
      myservo.write(pos);
      delay(stepDelay);
    }
  }
}

void loop() {
  digitalWrite(PIN_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIG, LOW);

  temp = float(pulseIn(PIN_ECHO, HIGH));
  cm = (temp * 17) / 1000;

  Serial.print("Echo = ");
  Serial.print(temp);
  Serial.print(",  Distance = ");
  Serial.print(cm);
  Serial.println("cm");

  int currentPos = myservo.read();
  if (cm < 30) {
    gradualMove(currentPos, 0); // 0度为舵机的顺时针方向
  } else {
    gradualMove(currentPos, 270); // 270度为舵机的初始位置
  }

  delay(100);
}
```

输出

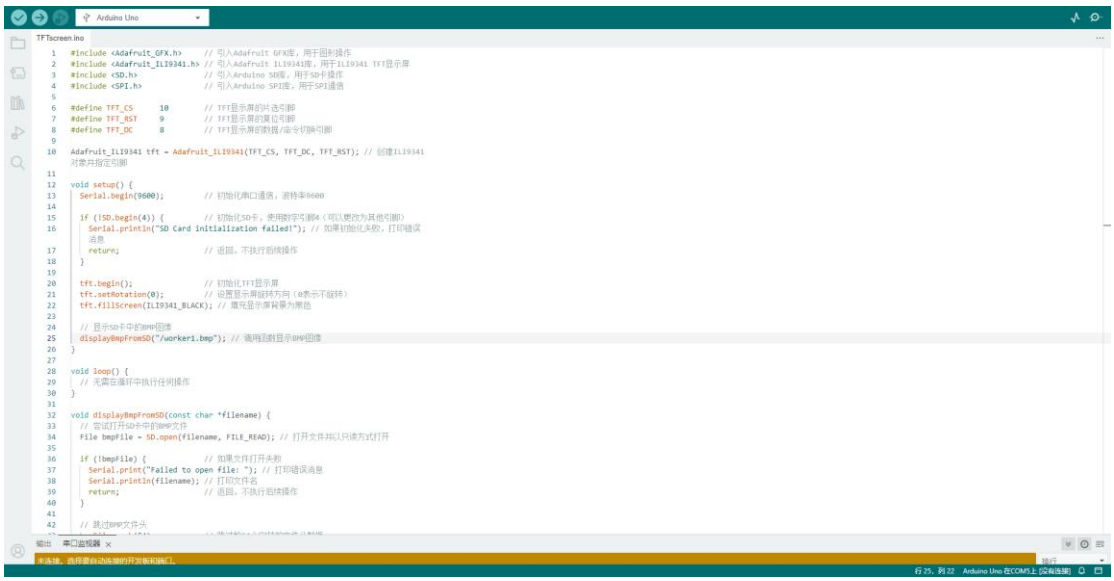
行 31，列 4　Arduino Leonardo 在COM6上 [没有连接]

Ardunio distance & sevor code



```
#include "Adafruit_TCS34725.h"

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_614MS,
TCS34725_GAIN_1X);
const int ledPin = 10; // 连接LED的数字引脚

int previousGreenValue = 0; // 声明并初始化 previousGreenValue
int threshold = 10; // 声明并初始化 threshold

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

  if (tcs.begin()) {
    Serial.println("Found sensor");
  } else {
    Serial.println("No TCS34725 found ... check your connections");
    while (1);
  }

  pinMode(ledPin, OUTPUT); // 设置LED引脚为输出
  digitalWrite(ledPin, LOW); // 初始状态下熄灭LED
}

void loop(void) {
  uint16_t r, g, b, c;
  tcs.getRawData(&r, &g, &b, &c);

  // 打印RGB通道值
  Serial.print("R: "); Serial.print(r, DEC); Serial.print(" ");
  Serial.print("G: "); Serial.print(g, DEC); Serial.print(" ");
  Serial.print("B: "); Serial.print(b, DEC); Serial.print(" ");
  Serial.print("C: "); Serial.print(c, DEC); Serial.print(" ");

  // 检测绿色通道值的变化
  if (abs(g - previousGreenValue) > threshold) {
    if (g > previousGreenValue) {
      // 绿色通道值增加，点亮LED
      digitalWrite(ledPin, HIGH); // 将LED引脚设置为高电平，点亮LED
    } else {
      // 绿色通道值减小，熄灭LED
      digitalWrite(ledPin, LOW); // 将LED引脚设置为低电平，熄灭LED
    }
    previousGreenValue = g;
  }
```

Rgb color sensor code



```
#include <Adafruit_GFX.h>     // 引入Adafruit GFX库，用于图形操作
#include <Adafruit_ILI9341.h> // 引入Adafruit ILI9341库，用于ILI9341 TFT显示屏
#include <SD.h>               // 引入Arduino SD库，用于SD卡操作
#include <SPI.h>              // 引入Arduino SPI库，用于SPI通信

#define TFT_CS      10        // TFT显示屏的片选引脚
#define TFT_RST     9         // TFT显示屏的复位引脚
#define TFT_DC      8         // TFT显示屏的数据/命令切换引脚

Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC, TFT_RST); // 创建ILI9341
对象并指定引脚

void setup() {
  Serial.begin(9600);         // 初始化串口通信，波特率9600

  if (!SD.begin(4)) {         // 初始化SD卡，使用数字引脚4（可以更改为其他引脚）
    Serial.println("SD Card initialization failed!"); // 如果初始化失败，打印错误
    消息
    return;                   // 返回，不执行后续操作
  }

  tft.begin();                // 初始化TFT显示屏
  tft.setRotation(0);         // 设置显示屏旋转方向（0表示不旋转）
  tft.fillScreen(ILI9341_BLACK); // 填充显示屏背景为黑色

  // 显示SD卡中的BMP图像
  displayBmpFromSD("/worker1.bmp"); // 调用函数显示BMP图像
}

void loop() {
  // 无需在循环中执行任何操作
}

void displayBmpFromSD(const char *filename) {
  // 尝试打开SD卡中的BMP文件
  File bmpFile = SD.open(filename, FILE_READ); // 打开文件并以只读方式打开

  if (!bmpFile) {             // 如果文件打开失败
    Serial.print("Failed to open file: "); // 打印错误消息
    Serial.println(filename); // 打印文件名
    return;                   // 返回，不执行后续操作
  }

  // 跳过BMP文件头
```

```
}

void loop() {
  // 无需在循环中执行任何操作
}

void displayBmpFromSD(const char *filename) {
  // 尝试打开SD卡中的BMP文件
  File bmpFile = SD.open(filename, FILE_READ); // 打开文件并以只读方式打开

  if (!bmpFile) {             // 如果文件打开失败
    Serial.print("Failed to open file: "); // 打印错误消息
    Serial.println(filename); // 打印文件名
    return;                   // 返回，不执行后续操作
  }

  // 跳过BMP文件头
  bmpFile.seek(54);           // 跳过前54个字节的文件头数据

  // 读取BMP图像的宽度和高度
  uint32_t bmpWidth = bmpFile.read() + (bmpFile.read() << 8) + (bmpFile.read()
  << 16) + (bmpFile.read() << 24); // 读取宽度数据
  uint32_t bmpHeight = bmpFile.read() + (bmpFile.read() << 8) + (bmpFile.read()
  << 16) + (bmpFile.read() << 24); // 读取高度数据

  // 读取BMP图像数据并在TFT屏幕上显示
  for (int y = bmpHeight - 1; y >= 0; y--) { // 从底部向上遍历图像行
    for (int x = 0; x < bmpWidth; x++) {     // 从左到右遍历图像列
      uint8_t b = bmpFile.read();            // 读取蓝色分量
      uint8_t g = bmpFile.read();            // 读取绿色分量
      uint8_t r = bmpFile.read();            // 读取红色分量

      // 当RGB值以设置像素颜色
      uint16_t color = tft.color565(r, g, b); // 根据RGB值创建颜色

      // 在TFT屏幕上绘制像素
      tft.drawPixel(x, y, color);            // 在指定位置绘制像素
    }
  }

  // 关闭文件
  bmpFile.close();                          // 关闭文件
}
```

TFT screen code