

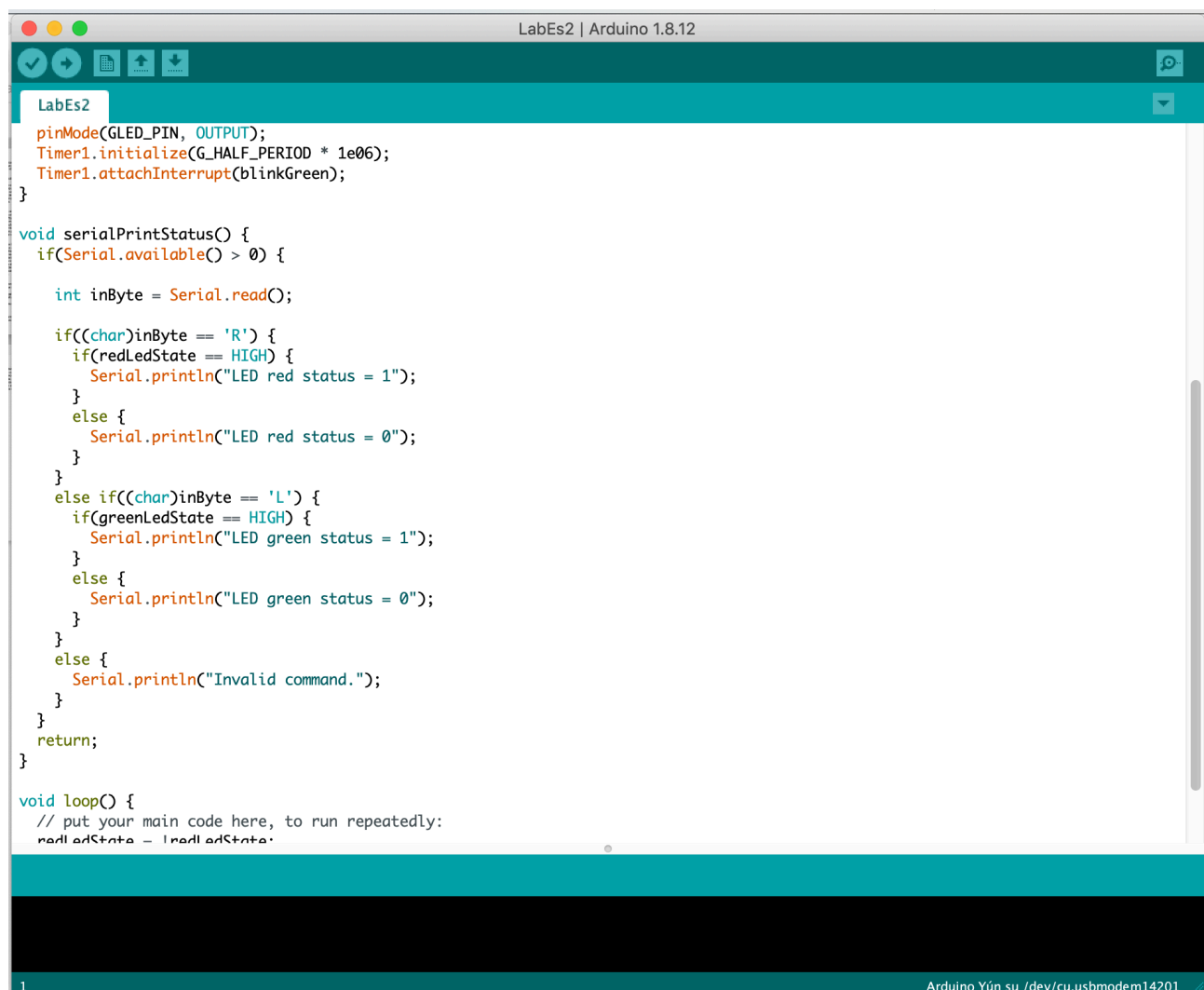
Eugenio Principi - s236654  
Arya Houshmand - s247275  
Leonardo Perugini - s249807

## Esercizio 1.2

Lo scheletro del codice, che detta il comportamento dei led è identico all'esercizio 1.

La loro accensione e funzionamento però, avviene solo al momento in cui l'utente desidera inserire input, la cui gestione è delegata alla funzione serialPrintStatus, che a seconda del carattere inserito (R = led rosso, L = led verde), informa l'utente riguardo allo stato (acceso o meno) del led corrispondente.

Se invece l'utente inserisce un carattere non abilitato viene stampato un messaggio d'errore.



```
LabEs2 | Arduino 1.8.12

pinMode(GLED_PIN, OUTPUT);
Timer1.initialize(G_HALF_PERIOD * 1e06);
Timer1.attachInterrupt(blinkGreen);
}

void serialPrintStatus() {
  if(Serial.available() > 0) {

    int inByte = Serial.read();

    if((char)inByte == 'R') {
      if(redLedState == HIGH) {
        Serial.println("LED red status = 1");
      }
      else {
        Serial.println("LED red status = 0");
      }
    }
    else if((char)inByte == 'L') {
      if(greenLedState == HIGH) {
        Serial.println("LED green status = 1");
      }
      else {
        Serial.println("LED green status = 0");
      }
    }
    else {
      Serial.println("Invalid command.");
    }
  }
  return;
}

void loop() {
  // put your main code here, to run repeatedly:
  redLedState = !redLedState;
}
```

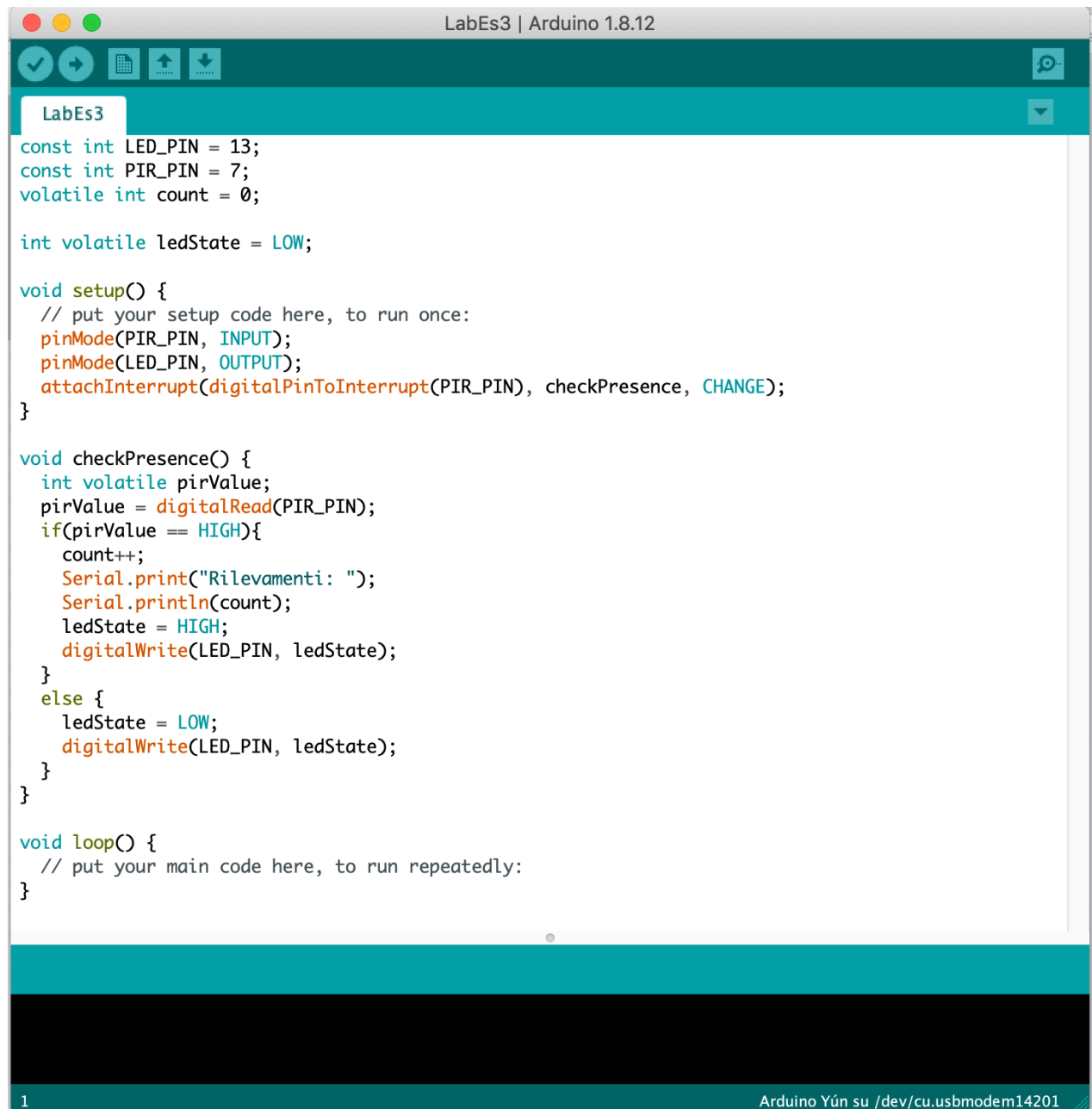
## Esercizio 1.3

In questo esercizio si sfruttava un sensore di movimento ad infrarossi per contare il numero di persone che passavano davanti ad esso.

La funzione checkPresence (che controlla lo stato del sensore e aggiorna il numero di passaggi) è chiamata solo quando si verifica un movimento, tramite la ISR

designata con *attachInterrupt*, nello specifico, solo al momento in cui il valore del pin associato al sensore subisce una variazione.

Inoltre nel circuito è presente un led che viene attivato solo al passaggio di qualcuno davanti al sensore.



```
LabEs3 | Arduino 1.8.12

const int LED_PIN = 13;
const int PIR_PIN = 7;
volatile int count = 0;

int volatile ledState = LOW;

void setup() {
  // put your setup code here, to run once:
  pinMode(PIR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(PIR_PIN), checkPresence, CHANGE);
}

void checkPresence() {
  int volatile pirValue;
  pirValue = digitalRead(PIR_PIN);
  if(pirValue == HIGH){
    count++;
    Serial.print("Rilevamenti: ");
    Serial.println(count);
    ledState = HIGH;
    digitalWrite(LED_PIN, ledState);
  }
  else {
    ledState = LOW;
    digitalWrite(LED_PIN, ledState);
  }
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

1 Arduino Yún su /dev/cu.usbmodem14201

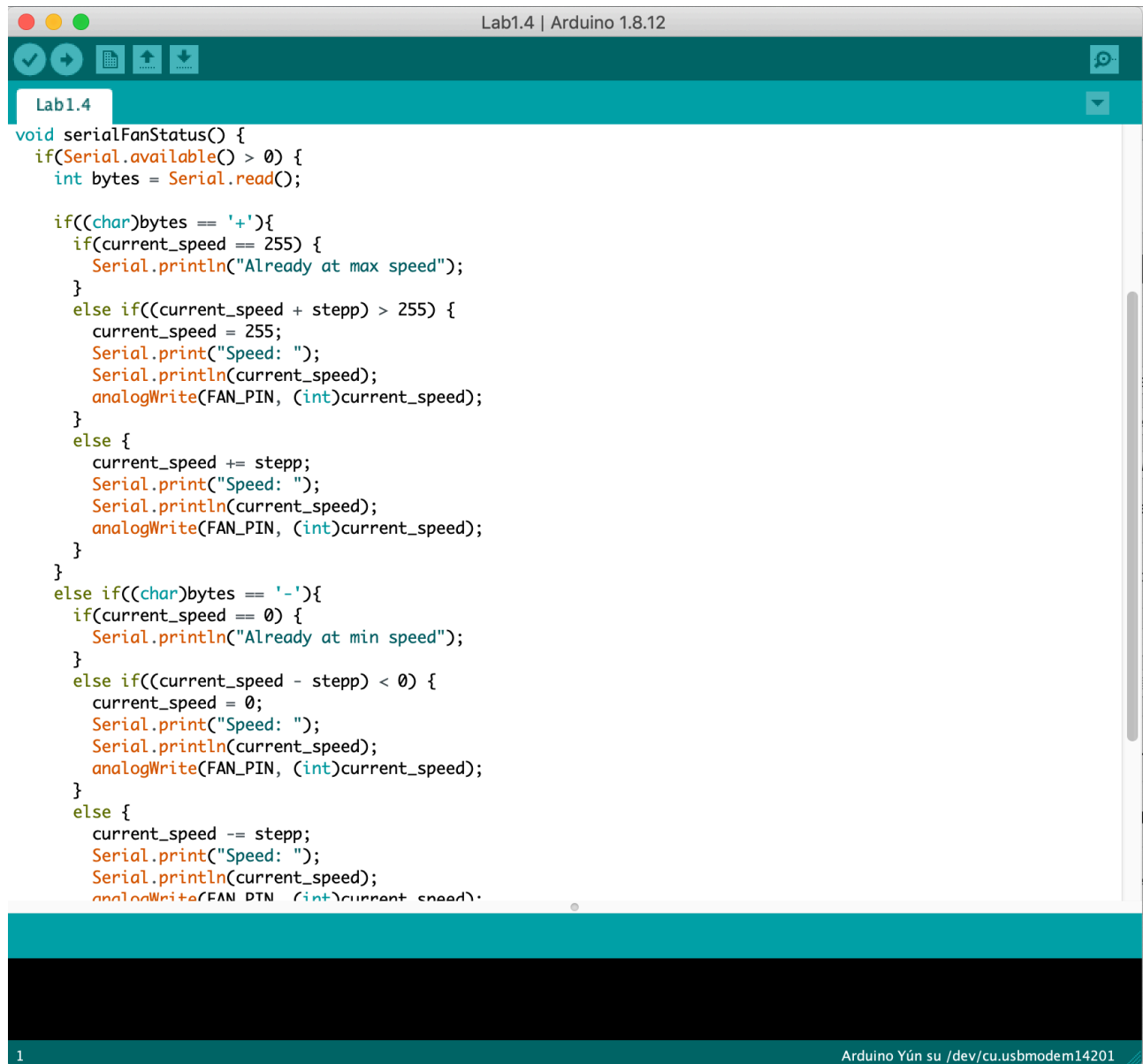
## Esercizio 1.4

Questo esercizio ha lo stesso scheletro logico del 1.2.

Alla scheda Arduino è collegato un piccolo motore con un'elica alimentato a corrente continua tramite PWM.

La velocità di rotazione della ventola è a discrezione dell'utente, che con l'inserimento dei caratteri +/- può aumentare di uno step uguale a 10, per un massimo di 255, la velocità. Analogamente, se viene inserito il carattere '-', la velocità diminuisce di 10.

Sono stati inseriti dei meccanismi di controllo per evitare che la velocità inserita sfiori dal range consentito (0 - 255), in questo caso infatti l'utente verrà notificato con un messaggio.



```
Lab1.4 | Arduino 1.8.12

void serialFanStatus() {
  if(Serial.available() > 0) {
    int bytes = Serial.read();

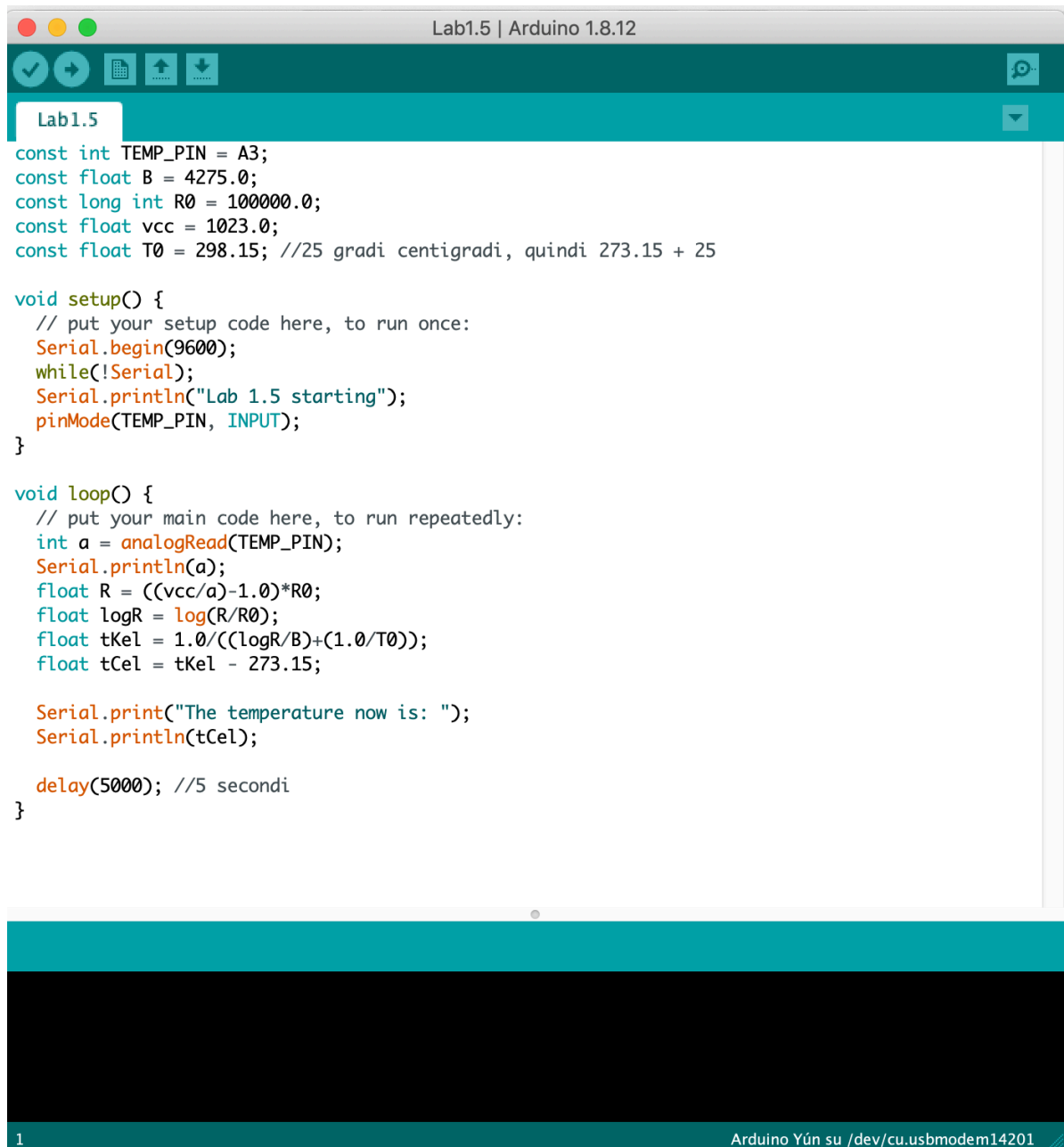
    if((char)bytes == '+'){
      if(current_speed == 255) {
        Serial.println("Already at max speed");
      }
      else if((current_speed + stepp) > 255) {
        current_speed = 255;
        Serial.print("Speed: ");
        Serial.println(current_speed);
        analogWrite(FAN_PIN, (int)current_speed);
      }
      else {
        current_speed += stepp;
        Serial.print("Speed: ");
        Serial.println(current_speed);
        analogWrite(FAN_PIN, (int)current_speed);
      }
    }
    else if((char)bytes == '-'){
      if(current_speed == 0) {
        Serial.println("Already at min speed");
      }
      else if((current_speed - stepp) < 0) {
        current_speed = 0;
        Serial.print("Speed: ");
        Serial.println(current_speed);
        analogWrite(FAN_PIN, (int)current_speed);
      }
      else {
        current_speed -= stepp;
        Serial.print("Speed: ");
        Serial.println(current_speed);
        analogWrite(FAN_PIN, (int)current_speed);
      }
    }
  }
}
```

## Esercizio 1.5

L'esercizio prevedeva la rilevazione della temperatura tramite Thermistor. Grazie al partitore di tensione interno al sensore, la temperatura misurata è sufficientemente precisa.

Seguendo i passaggi presentati nelle slide è stato possibile scrivere i calcoli per ottenere la temperatura in maniera corretta, prima in gradi Kelvin e poi convertita in gradi Celsius.

La temperatura misurata viene poi stampata sul monitor seriale ad intervalli di 5 secondi.



```
Lab1.5 | Arduino 1.8.12

const int TEMP_PIN = A3;
const float B = 4275.0;
const long int R0 = 100000.0;
const float vcc = 1023.0;
const float T0 = 298.15; //25 gradi centigradi, quindi 273.15 + 25

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  while(!Serial);
  Serial.println("Lab 1.5 starting");
  pinMode(TEMP_PIN, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  int a = analogRead(TEMP_PIN);
  Serial.println(a);
  float R = ((vcc/a)-1.0)*R0;
  float logR = log(R/R0);
  float tKel = 1.0/((logR/B)+(1.0/T0));
  float tCel = tKel - 273.15;

  Serial.print("The temperature now is: ");
  Serial.println(tCel);

  delay(5000); //5 secondi
}
```

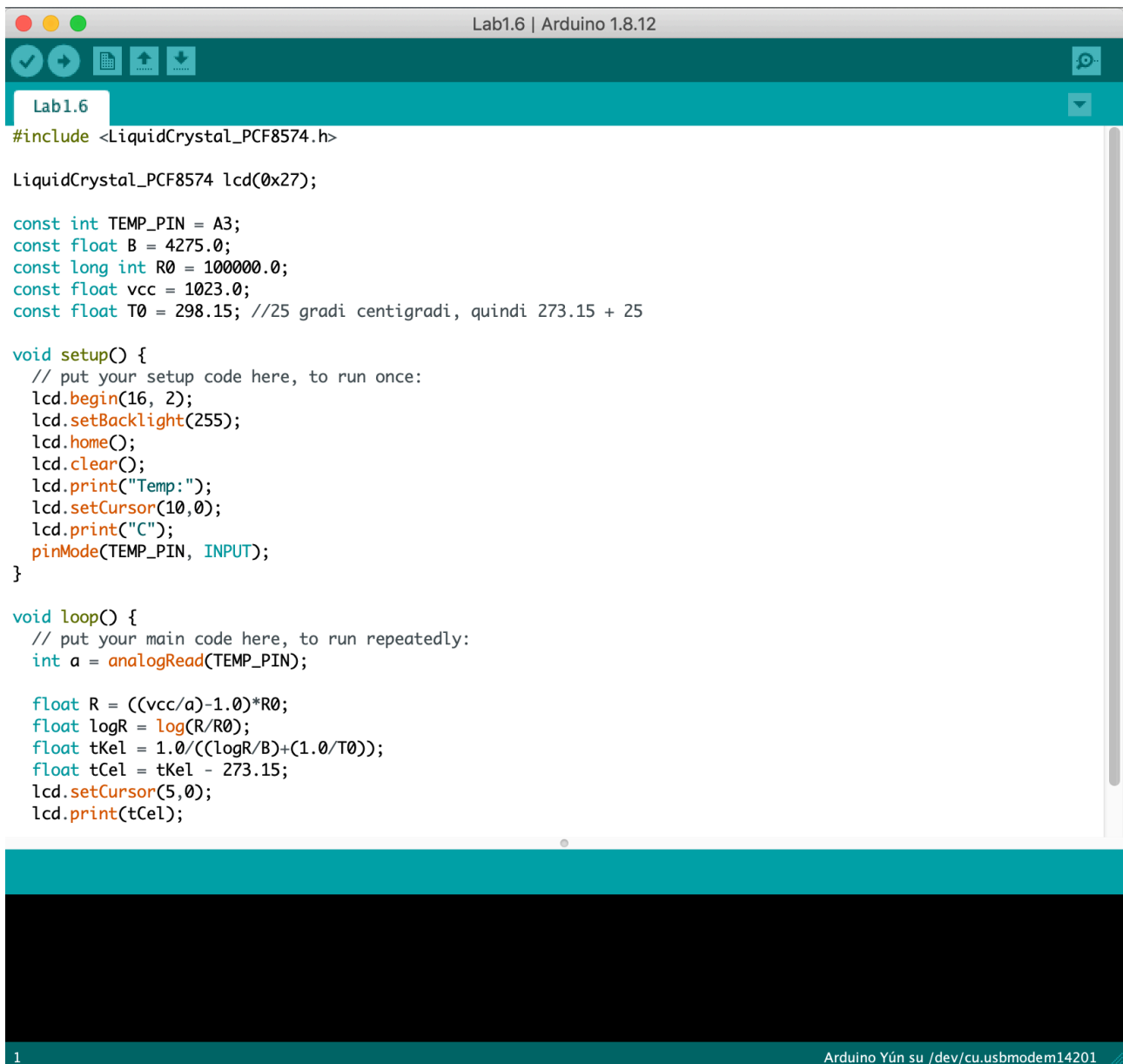
1 Arduino Yún su /dev/cu.usbmodem14201

## Esercizio 1.6

Trattasi di prolungamento naturale dell'esercizio 1.5, in cui tuttavia la temperatura è rappresentata su uno schermo LCD.

In questo caso è stato necessario importare una libreria dedicata Arduino per potere comunicare con lo schermo, tramite protocollo I2C.

Per ridurre al minimo i dati trasportati sul bus I2C, si è deciso di mantenere come stampa iniziale sullo schermo le parole "Temp: " e "C", che sono infatti scritte all'interno della funzione di setup. Dopodiché abbiamo inserito nell'opportuna posizione, grazie al metodo setCursor la temperatura misurata, anche in questo caso aggiornandola ogni 5 secondi.



The screenshot shows the Arduino IDE interface with a file named 'Lab1.6'. The code is written in C++ and uses the LiquidCrystal\_PCF8574 library. It defines constants for the temperature sensor (A3), the beta value (4275.0), the resistor value (100000.0), the VCC voltage (1023.0), and the reference temperature (298.15). The setup function initializes the LCD, sets the backlight, and configures the temperature pin as an input. The loop function reads the analog value from the temperature pin, calculates the resistance, and then the temperature in Kelvin and Celsius, which are then displayed on the LCD.

```
Lab1.6 | Arduino 1.8.12

#include <LiquidCrystal_PCF8574.h>

LiquidCrystal_PCF8574 lcd(0x27);

const int TEMP_PIN = A3;
const float B = 4275.0;
const long int R0 = 100000.0;
const float vcc = 1023.0;
const float T0 = 298.15; //25 gradi centigradi, quindi 273.15 + 25

void setup() {
  // put your setup code here, to run once:
  lcd.begin(16, 2);
  lcd.setBacklight(255);
  lcd.home();
  lcd.clear();
  lcd.print("Temp:");
  lcd.setCursor(10,0);
  lcd.print("C");
  pinMode(TEMP_PIN, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  int a = analogRead(TEMP_PIN);

  float R = ((vcc/a)-1.0)*R0;
  float logR = log(R/R0);
  float tKel = 1.0/((logR/B)+(1.0/T0));
  float tCel = tKel - 273.15;
  lcd.setCursor(5,0);
  lcd.print(tCel);
}
```

1 Arduino Yún su /dev/cu.usbmodem14201