Summer School

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Week 1 Task submission

Prajwal M **Mechatronics core**

Department of Electrical Engineering-ICDT
Indian Institute of Technology Hyderabad
June 10, 2024

Contents

1 | Task 1 - Temperature control system using Matlab

1.1 | Simulink model

In this subsection, I illustrate the Simulink model for the temperature control system (Figure 1.1). This model includes various components such as the heater and the temperature display, which are essential for simulating the temperature control process. By visualizing this model, I can better understand how each component interacts within the system.

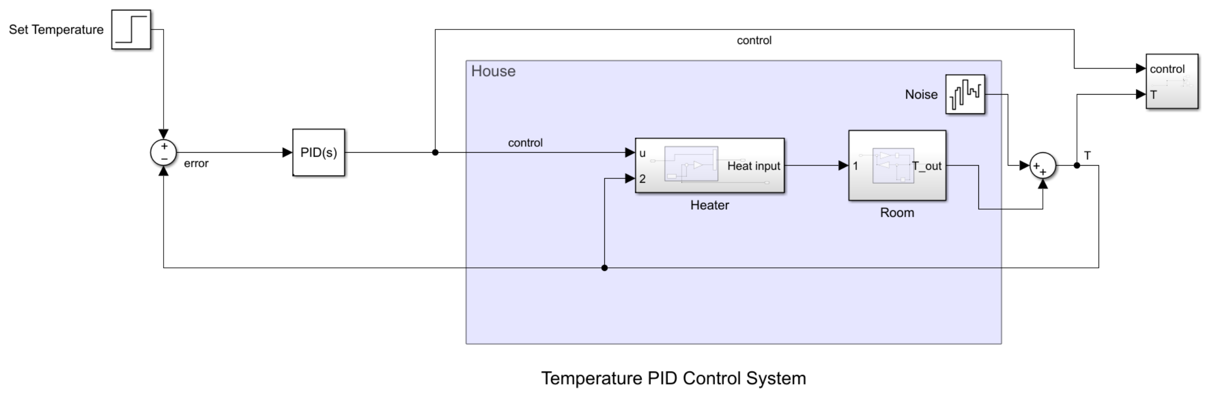


Figure 1.1: Temperature-control system

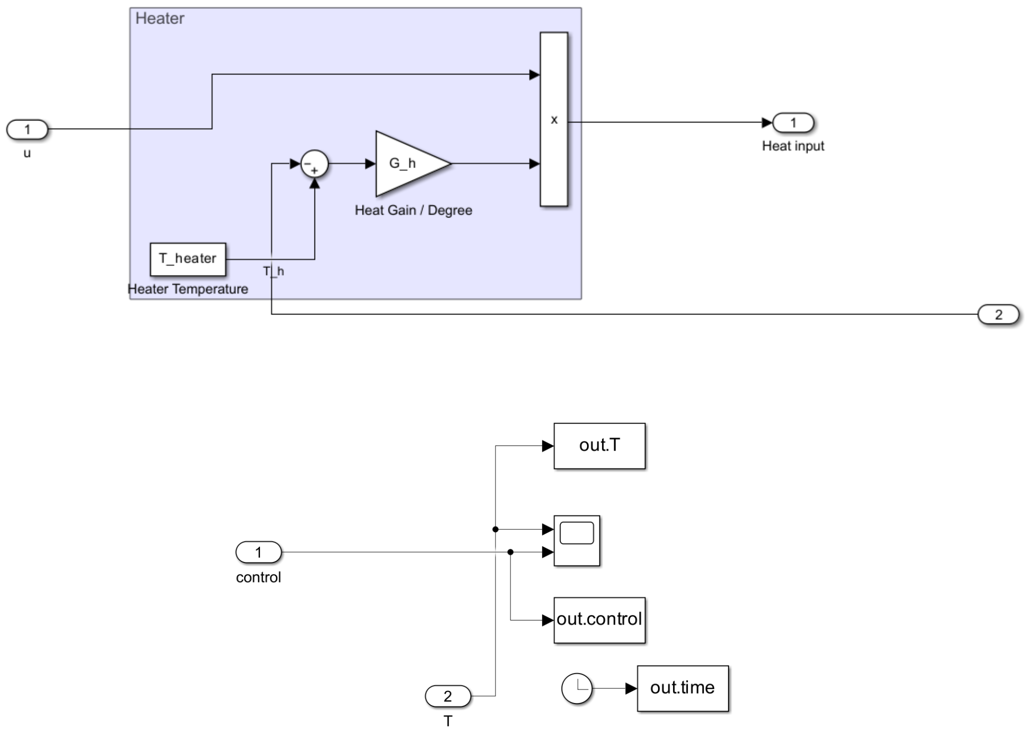


Figure 1.2: Heater and display

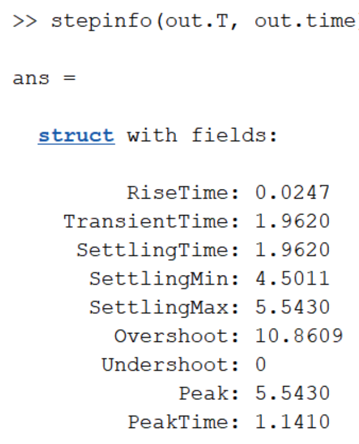
1.2 | Matlab code

In this part, I present the Matlab code used to control the temperature system. It defines the PID controller parameters, the heater's characteristics, and the room dynamics. This script is crucial for simulating the temperature control process and achieving the desired temperature setpoint. By running this code, I can observe the system's behavior and fine-tune the parameters as needed.

```matlab
clear
clc
close all

%-------------------------------
Ki = 5.5; %Integral control
Kp = 0.01; %Proportional control
Kd = 0.012; %Derivative control
%-------------------------------

Ts = 0.1; %Sampling rate of noise
Np = [0.001]; %noise control
setTemp = 5; %in degree celsius
G_h = 3600*3000; %heater constant
T_heater = 70; %temprature of heater
K2 = 1/(1470*1005.4); %room dynamics coefficent1
K1 = 1/4.285e-7; %room dynamics coefficeint2


GP_num = [G_h*K1*K2];
GP_den = [1, K1*K2];

GP = tf(GP_num, GP_den); %heater plant transfer function
```

1.3 | Controller Analysis

This subsection contains my analysis of the PID controller used in the temperature control system. Figure 1.3 shows the details of the PID controller settings, while Figure 1.4 illustrates the system's response to a unit-step input. This helps me understand the performance of the system in reaching and maintaining the desired temperature. Evaluating these results allows me to assess the controller's effectiveness and make necessary adjustments.



```
>> stepinfo(out.T, out.time)

ans =

  struct with fields:

        RiseTime: 0.0247
   TransientTime: 1.9620
    SettlingTime: 1.9620
     SettlingMin: 4.5011
     SettlingMax: 5.5430
       Overshoot: 10.8609
      Undershoot: 0
            Peak: 5.5430
        PeakTime: 1.1410
```

Figure 1.3: PID controller info

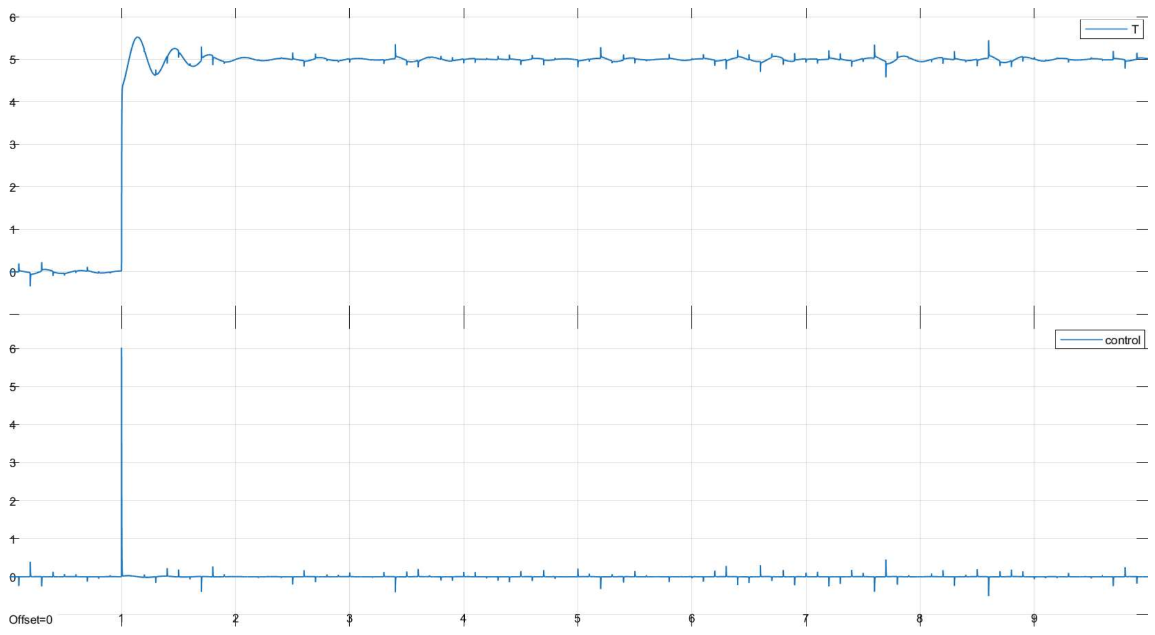


Figure 1.4: Unit-step response

1.4 | Learning outcome

Through this task, I have learned how to design and implement a temperature control system using Matlab and Simulink. The experience with PID controllers has been invaluable in understanding their impact on system stability and performance.

- Developed skills in using Matlab and Simulink for system modeling.
- Enhanced understanding of PID controllers and their tuning.
- Gained experience in designing and simulating control systems.

2 | Task 2 - Multi-sensor system

2.1 | TinkerCad model

In this subsection, I present the TinkerCad model representing a multi-sensor system circuit (Figure 2.1). This schematic diagram includes all the connections and components required for the system, providing a visual guide for building the actual hardware setup. By examining this model, I can ensure all components are correctly placed and connected.

2.2 | Arduino code

In this part, I provide the Arduino code responsible for detecting the presence of a person using multiple sensors and displaying the information on an LCD screen. This script is fundamental for the operation of the multi-sensor system, integrating sensor data and outputting the results. Running this code allows me to see how the system responds to real-world inputs.

```
#include <Servo.h>
#include <LiquidCrystal.h>
```

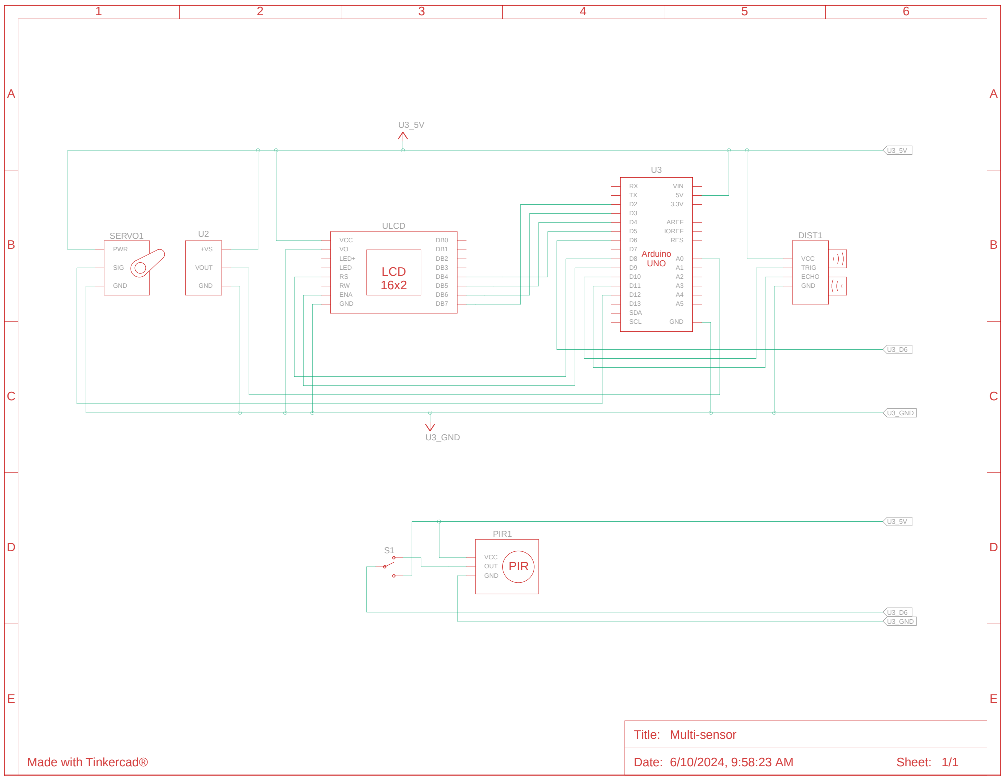
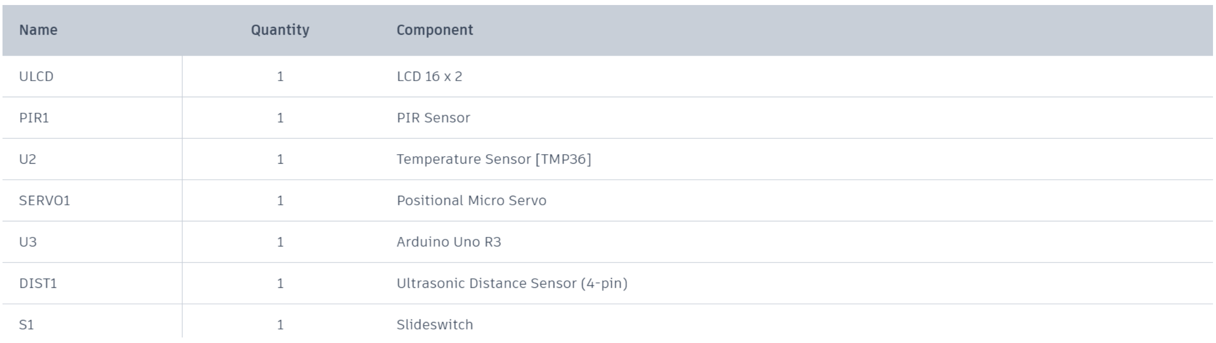


Figure 2.1: Circuit schematic diagram

```

#define trigPin 10
#define echoPin 11
#define pirPin 6
#define tempPin A0

Servo myservo;
LiquidCrystal lcd(8, 9, 5, 4, 3, 2);

bool personDetected = false;
float distanceBuffer;
bool detected;

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

  lcd.begin(16, 2);
  lcd.print("Initializing...");
  delay(2000);
  lcd.clear();

  myservo.attach(12);
  myservo.write(0);

  pinMode(trigPin, OUTPUT);
```



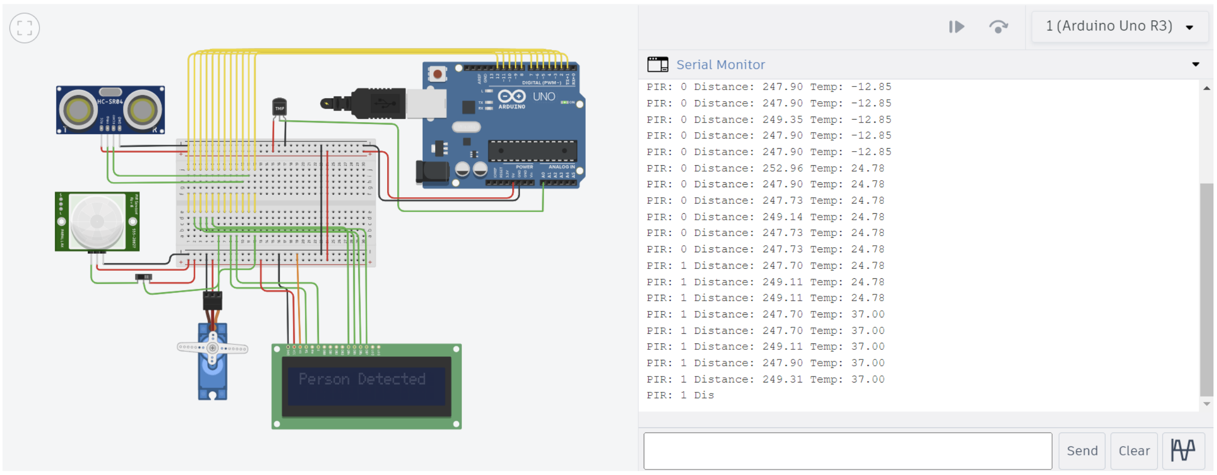
Name	Quantity	Component
ULCD	1	LCD 16 x 2
PIR1	1	PIR Sensor
U2	1	Temperature Sensor [TMP36]
SERVO1	1	Positional Micro Servo
U3	1	Arduino Uno R3
DIST1	1	Ultrasonic Distance Sensor (4-pin)
S1	1	Slideswitch

Figure 2.2: Components used

```
  pinMode(echoPin, INPUT);
  pinMode(pirPin, INPUT);


}

void loop() {
  int pirState = digitalRead(pirPin);
  float distance = readDistance();
  float temperatureC = readTemperature();

  bool personDetected = detectPerson(pirState, distance, temperatureC);

  displayStatus(personDetected);

  delay(1000);

  // Debug output
  Serial.print("PIR: ");
  Serial.print(pirState);
  Serial.print(" Distance: ");
  Serial.print(distance);
  Serial.print(" Temp: ");
  Serial.println(temperatureC);
}

float readDistance() {
  long duration;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  return (duration / 2) / 29.1;
}

float readTemperature() {
  int tempValue = analogRead(tempPin);
  float voltage = tempValue * (5.0 / 1023.0);
  return (voltage - 0.5) * 100.0;
}

bool detectPerson(int pirState, float distance, float temperatureC) {
  bool distanceReducing;
  bool temperatureInRange = (temperatureC > 30.0 && temperatureC < 40.0);

  distanceReducing = (distanceBuffer > distance);
  distanceBuffer = distance;

  if (pirState == HIGH && distanceReducing && temperatureInRange)
  {
    detected = true;
  }

  if (!distanceReducing && !temperatureInRange && pirState == LOW &&
     distance > 200)
  {
        detected = false;
  }
  return detected;
}

void displayStatus(bool personDetected) {
  lcd.clear();
  if (personDetected) {
    lcd.print("Person Detected");
    myservo.write(90);
  } else {
    lcd.print("No Person");
    myservo.write(0);
  }
}
```

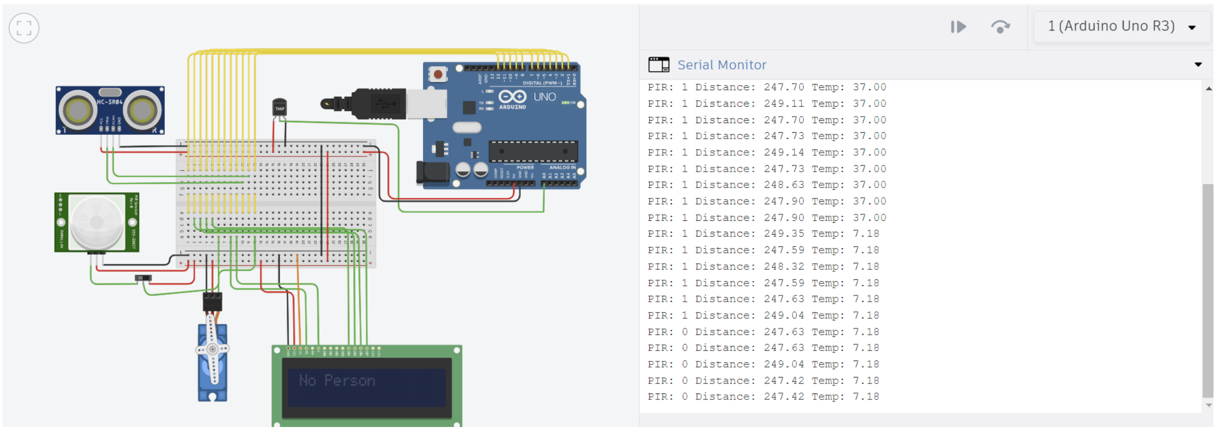
2.3 | Simulation

Figures 2.3 and 2.4 show the simulation running in TinkerCad. These images demonstrate the system in action, verifying that the circuit and code function correctly to detect a person and display the information accurately. Observing the simulation helps me confirm that the system works as intended and identify any necessary improvements.



```
PIR: 0 Distance: 247.90 Temp: -12.85
PIR: 0 Distance: 247.90 Temp: -12.85
PIR: 0 Distance: 249.35 Temp: -12.85
PIR: 0 Distance: 247.90 Temp: -12.85
PIR: 0 Distance: 247.90 Temp: -12.85
PIR: 0 Distance: 252.96 Temp: 24.78
PIR: 0 Distance: 247.90 Temp: 24.78
PIR: 0 Distance: 247.73 Temp: 24.78
PIR: 0 Distance: 249.14 Temp: 24.78
PIR: 0 Distance: 247.73 Temp: 24.78
PIR: 0 Distance: 247.73 Temp: 24.78
PIR: 1 Distance: 247.70 Temp: 24.78
PIR: 1 Distance: 249.11 Temp: 24.78
PIR: 1 Distance: 249.11 Temp: 24.78
PIR: 1 Distance: 247.70 Temp: 37.00
PIR: 1 Distance: 247.70 Temp: 37.00
PIR: 1 Distance: 249.11 Temp: 37.00
PIR: 1 Distance: 247.90 Temp: 37.00
PIR: 1 Distance: 249.31 Temp: 37.00
PIR: 1 Dis
```

Figure 2.3: Simulation running



Serial Monitor

```
PIR: 1 Distance: 247.70 Temp: 37.00
PIR: 1 Distance: 249.11 Temp: 37.00
PIR: 1 Distance: 247.70 Temp: 37.00
PIR: 1 Distance: 247.73 Temp: 37.00
PIR: 1 Distance: 249.14 Temp: 37.00
PIR: 1 Distance: 247.73 Temp: 37.00
PIR: 1 Distance: 248.63 Temp: 37.00
PIR: 1 Distance: 247.90 Temp: 37.00
PIR: 1 Distance: 247.90 Temp: 37.00
PIR: 1 Distance: 249.35 Temp: 7.18
PIR: 1 Distance: 247.59 Temp: 7.18
PIR: 1 Distance: 248.32 Temp: 7.18
PIR: 1 Distance: 247.59 Temp: 7.18
PIR: 1 Distance: 247.63 Temp: 7.18
PIR: 1 Distance: 249.04 Temp: 7.18
PIR: 0 Distance: 247.63 Temp: 7.18
PIR: 0 Distance: 247.63 Temp: 7.18
PIR: 0 Distance: 249.04 Temp: 7.18
PIR: 0 Distance: 247.42 Temp: 7.18
PIR: 0 Distance: 247.42 Temp: 7.18
```

Figure 2.4: Simulation running - second view

2.4 | Learning outcome

By completing this task, I have developed an understanding of multi-sensor systems and their applications. I learned how to integrate different sensors using Arduino, write the necessary code, and simulate the system to verify its functionality.

- Learned to design and simulate arduino system using TinkerCad
- Designes a multi-sensor system.
- Understood the process of verifying and troubleshooting sensor systems.