Arduino Custom Remote Control System Using Pulse-Duration Encoding

COURSE: SELECTED TOPICS IN CS

Presented To: Alaa Hamdy

Prepared By:

Roaa Khaled — ID: 2022/05885

Nouran Hassan — ID: 2022/00062

Contents

Abstract

This project presents the design and implementation of a custom remote control system using Arduino microcontrollers. The system transmits commands using pulse-duration–based signaling rather than standard infrared communication protocols. Commands are generated using push buttons on a transmitter Arduino and decoded on a receiver Arduino to control RGB LEDs. The system was validated through both hardware implementation and a software-based simulator.

1. Introduction

Remote control systems are widely used in embedded and IoT applications. Most commercial systems rely on standardized infrared protocols; however, this project focuses on understanding low-level communication by implementing a **custom timing-based protocol**. The project emphasizes signal generation, timing measurement, and decoding using Arduino digital I/O.

2. Project Objectives

- Implement a custom communication protocol using pulse durations
- Design transmitter and receiver Arduino systems
- Decode commands without using IR libraries
- Verify system behavior using a simulator
- Demonstrate reliable LED control based on transmitted commands

3. System Overview

The system consists of two main units:

- **Transmitter Unit:** Reads button inputs and generates pulse-duration signals
- **Receiver Unit:** Measures incoming pulse duration and activates corresponding LEDs

4. Components Used

Component	Quantity	Description
Arduino Uno	2	Transmitter and Receiver
Push Buttons	3	User input
LEDs / RGB LED	1 RGB / 3 LEDs	Output indicators
220 Ohm Resistor	7	Current limiting
Jumper Wires	17	Connections
Breadboard	2	Circuit assembly
IR LED 5mm Infrared Transmitter	1	IR signal detection on receiver side
TCRT5000 Infrared Line Tracking Sensor Module	1	IR signal transmission

5. Transmitter Design

5.1 Hardware Configuration

Arduino Uno reads three push buttons (pins 4, 5, 6). A 5mm IR LED connected to pin 3 emits pulse-duration signals corresponding to each button.5.2 Signal Encoding Strategy

- Button 1 → 200 ms single pulse
- Button 2 → 400 ms single pulse
- Button 3 → 600 ms single pulse

This approach ensures unique and easily distinguishable timing patterns for each command.

5.3 Transmitter Code

```
void setup() {
  pinMode(3, OUTPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Button 1 - Single 200ms pulse
  if (digitalRead(4) == HIGH) {
    Serial.println("B1 - 200ms");
    digitalWrite(3, HIGH);
    delay(200);
    digitalWrite(3, LOW);
    while(digitalRead(4) == HIGH);
  }

  // Button 2 - Single 400ms pulse
  if (digitalRead(5) == HIGH) {
    Serial.println("B2 - 400ms");
    digitalWrite(3, HIGH);
    delay(400);
    digitalWrite(3, LOW);
    while(digitalRead(5) == HIGH);
  }

  // Button 3 - Single 600ms pulse
  if (digitalRead(6) == HIGH) {
    Serial.println("B3 - 600ms");
    digitalWrite(3, HIGH);
    delay(600);
    digitalWrite(3, LOW);
    while(digitalRead(6) == HIGH);
  }
}
```

6. Receiver Design

6.1 Hardware Configuration

TCRT5000 sensor input connected to Arduino detects transmitted pulses. RGB LED connected to output pins displays the received command.

6.2 Signal Decoding Strategy

Arduino measures the LOW pulse duration and compares it to predefined thresholds:

- 150–250 ms → RED LED (Button 1)
- 350–450 ms → GREEN LED (Button 2)
- 550–650 ms → BLUE LED (Button 3)

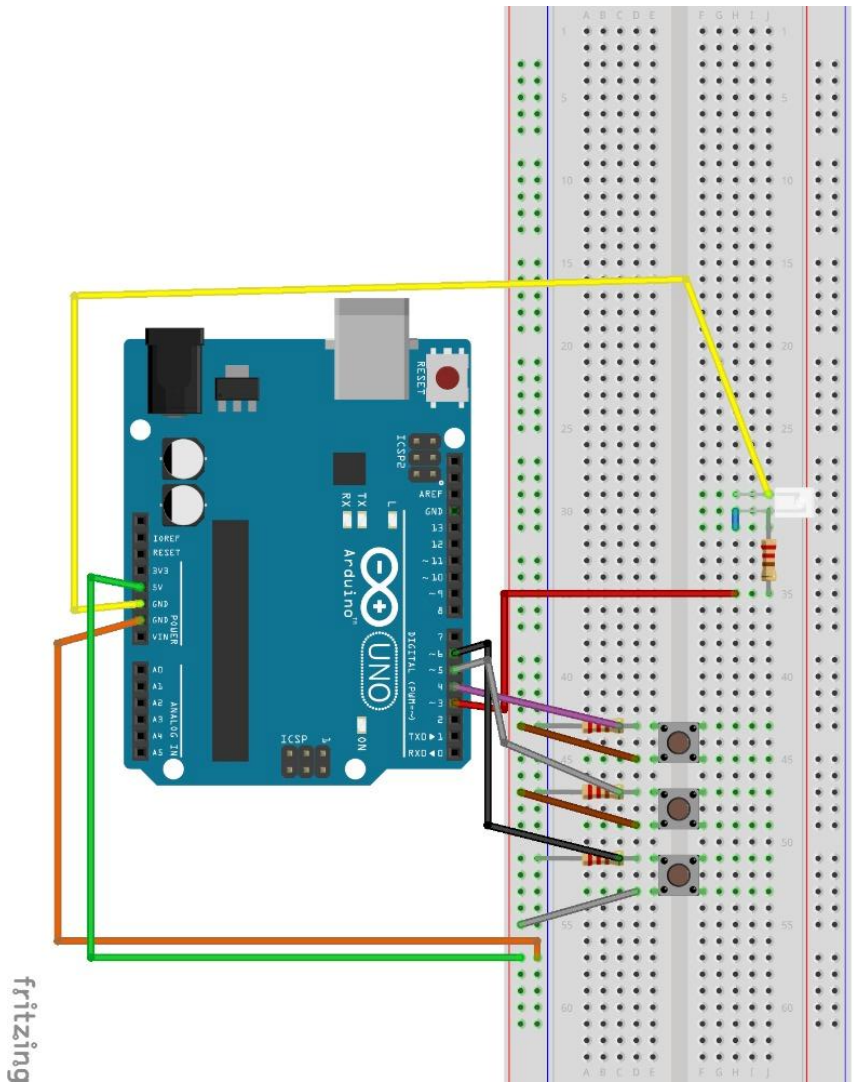
6.3 Receiver Code

```
void setup() {
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);  // Red
  pinMode(4, OUTPUT);  // Green
  pinMode(5, OUTPUT);  // Blue
  Serial.begin(9600);
  Serial.println("READY");
}

void loop() {
  if (digitalRead(2) == LOW) {
    unsigned long start = millis();

    // Wait for signal to end (with timeout)
    while(digitalRead(2) == LOW && millis() - start < 2000);

    unsigned long pulseTime = millis() - start;

    Serial.print("Got: ");
    Serial.print(pulseTime);
    Serial.println(" ms");

    // NON-OVERLAPPING ranges:
    if (pulseTime > 150 && pulseTime < 250) {    // ~200ms
      Serial.println("B1 - RED");
      digitalWrite(3, HIGH); delay(2000); digitalWrite(3, LOW);
    }
    else if (pulseTime > 350 && pulseTime < 450) { // ~400ms
      Serial.println("B2 - GREEN");
      digitalWrite(4, HIGH); delay(2000); digitalWrite(4, LOW);
    }
    else if (pulseTime > 550 && pulseTime < 650) { // ~600ms
      Serial.println("B3 - BLUE");
      digitalWrite(5, HIGH); delay(2000); digitalWrite(5, LOW);
    }
    else {
      Serial.println("Noise - Ignored");
    }
  }
}
```

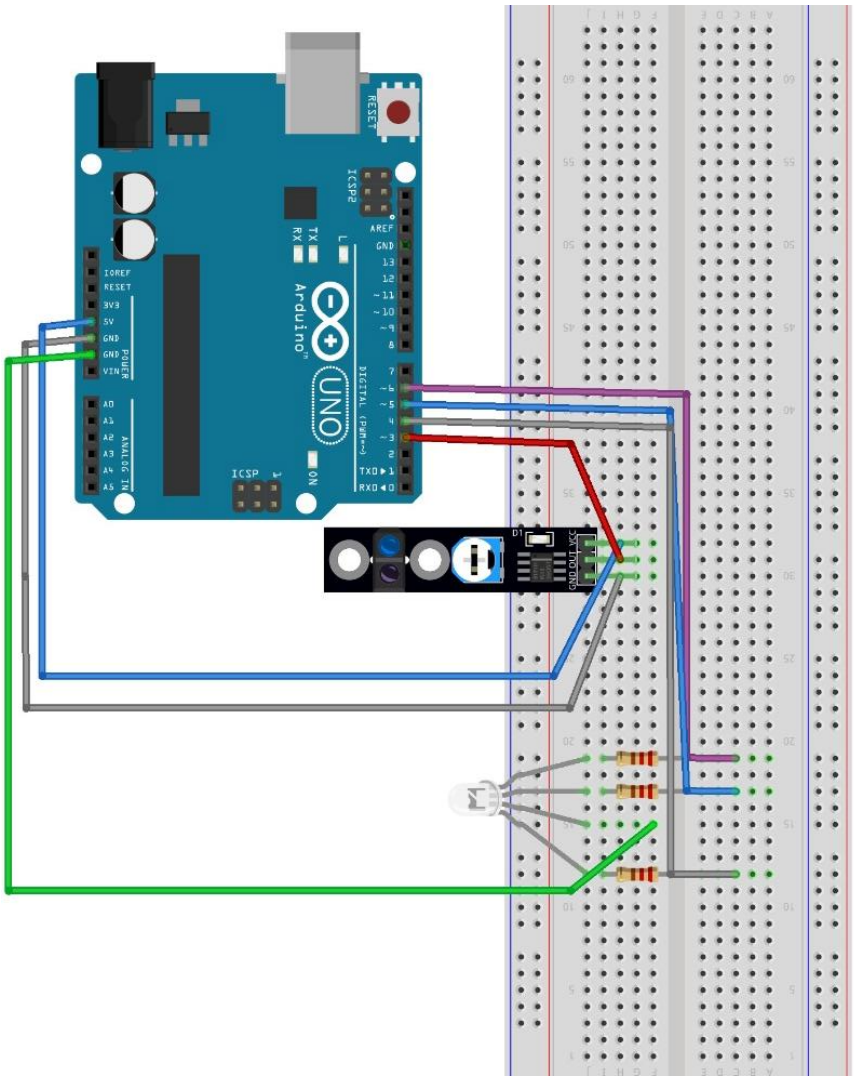
7.Simulator Design

A software-based simulator was created to emulate button presses and pulse-duration signals. This allowed verification of encoding and decoding logic without using physical hardware.

7.1 Transmitter



7.2 Receiver



8. Conclusion

This project successfully demonstrates a custom Arduino-based remote control system using pulse-duration encoding. It provided practical insight into low-level communication, timing analysis, and embedded system debugging.