CONTROL DE AFORO DEL CABLE AÉREO

Autor: Bryan Cartagena Hincapie
Sergio Henao Arbeláez

Ingeniería en Sistemas y Computación, Universidad de Caldas

Fundamentos de Internet de las Cosas

Dr.Diana Rocío Varón Serna

Tabla de Contenido

1. Introducción

El Internet de las Cosas (IoT) ha revolucionado la gestión de sistemas de transporte urbano, permitiendo el monitoreo en tiempo real y la optimización de recursos en contextos de movilidad masiva. Estas tecnologías resultan especialmente valiosas en sistemas de transporte como el cable aéreo, donde la seguridad y eficiencia dependen directamente del control de aforo en cada vagón.

El presente trabajo desarrolla un prototipo de control de aforo para la Línea 3 del Cable Aéreo de Manizales, utilizando Arduino, sensores ultrasónicos, indicadores

luminosos y un sistema de alertas mediante Telegram. El sistema busca simular el conteo automatizado de pasajeros que ingresan y salen del vagón, mediante dos sensores estratégicamente ubicados, complementado con señales visuales de estado y notificaciones remotas.

Con este proyecto se pretende demostrar la viabilidad de implementar soluciones IoT en el sistema de transporte masivo de la ciudad, mejorando la seguridad mediante la prevención de sobrecupo y proporcionando información en tiempo real para la toma de decisiones operativas.

2. Objetivos

General

Implementar un sistema IoT de control de aforo para el vagón del Cable Aéreo Línea 3 de Manizales que integre hardware (Arduino, sensores ultrasónicos, LEDs, buzzer) y software (Python, Telegram) para el monitoreo en tiempo real de la ocupación del vagón.

Específicos

1. Diseñar el esquema de ubicación de los dos sensores ultrasónicos en la entrada del vagón que permita diferenciar entre ingresos y salidas de pasajeros.

2. Programar la lógica de control en Arduino IDE que procese las lecturas de los sensores, actualice el contador de aforo y active los indicadores luminosos (LEDs) y sonoros (buzzer) según los umbrales establecidos.

3. Integrar el sistema físico con el bot de Telegram mediante un puente en Python que permita el envío de alertas automáticas y consultas remotas del estado de ocupación.

4. Validar el funcionamiento del prototipo mediante pruebas de conteo y comunicación, verificando la precisión del sistema y el envío oportuno de notificaciones.

3. Marco Teórico

3.1. Estado del arte y antecedentes

La implementación de sistemas IoT en el transporte masivo ha ganado relevancia como solución para optimizar la operación y garantizar la seguridad de los usuarios. En particular, el control de aforo se ha convertido en una necesidad crítica en sistemas de transporte como el cable aéreo, donde la capacidad limitada de los vagones requiere un monitoreo preciso en tiempo real.

El Internet de las Cosas (IoT) permite integrar componentes electrónicos de bajo costo con plataformas de comunicación digital, creando sistemas accesibles que pueden ser implementados en proyectos académicos. Estos desarrollos facilitan la comprensión de conceptos fundamentales como la adquisición de datos, el procesamiento de información y la comunicación hombre-máquina a través de interfaces modernas como Telegram.

3.2. Definiciones básicas

DIP switch: conjunto de interruptores mecánicos miniaturizados configurados en un solo módulo, que permiten simular entradas digitales al sistema. En este proyecto reemplazan la función de los sensores ultrasónicos para detectar el paso de personas.

LED (Light Emitting Diode): componente semiconductor que emite luz al ser polarizado directamente. Se emplea como indicador visual del estado de ocupación del vagón mediante tres colores: verde, amarillo y rojo.

Buzzer: transductor electroacústico que genera señales audibles al recibir un pulso eléctrico. Se utiliza para alertar cuando el vagón alcanza su capacidad máxima.

Arduino UNO: plataforma de desarrollo basada en el microcontrolador ATmega328P, que coordina todas las entradas y salidas del sistema físico.

Bot de Telegram: programa que ejecuta automáticamente tareas preconfiguradas dentro de la aplicación de mensajería, permitiendo la interacción remota con el sistema de aforo.

Sistema IoT: arquitectura que integra dispositivos físicos con servicios en la nube, permitiendo el monitoreo y control remoto a través de internet.

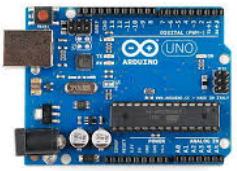
3.3. Desarrollo del producto

El prototipo implementado simula el sistema de control de aforo para un vagón del cable aéreo utilizando dos switches de un módulo DIP para representar los sensores de ingreso y salida. Cuando un switch se activa, el sistema interpreta el evento como el paso de una persona y actualiza el contador de ocupación.

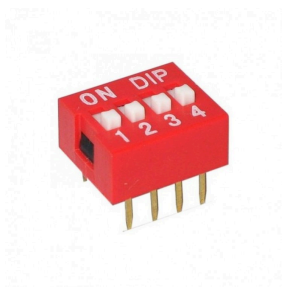
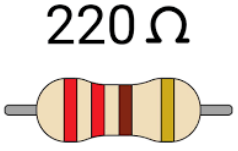
El sistema utiliza tres LEDs para indicar el estado de ocupación: verde para capacidad disponible, amarillo para ocupación media y rojo para capacidad completa. Al alcanzar el límite máximo, se activa el buzzer como alerta sonora local.

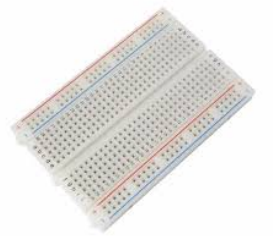
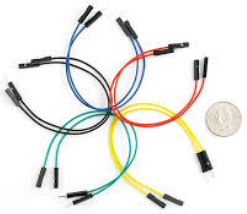
Paralelamente, mediante un puente desarrollado en Python, el sistema se comunica con un bot de Telegram que permite consultar el aforo actual en cualquier momento y recibe notificaciones automáticas cuando se superan los umbrales predefinidos, proporcionando así monitoreo remoto en tiempo real.

3.4. Tabla de componentes, herramientas e insumos

Nombre del componente	**Foto**	**Cantidad**	**Descripción / Función**
Arduino UNO		1	Placa de desarrollo que actúa como el cerebro del sistema, controlando sensores y actuadores.

ucaldas@ucaldas.edu.co www.ucaldas.edu.co PBX (57)(6) 878 15 00 Calle 65 # 26 - 10 | Manizales - Colombia

Módulo DIP switch		1	Contiene múltiples interruptores independientes; se utilizaron 2 switches para simular los sensores de ingreso y salida.
LED Rojo		1	Indica cuando el vagón ha alcanzado su capacidad máxima ("VAGÓN LLENO").
LED Amarillo		1	Señala que el vagón tiene ocupación media ("ÚLTIMOS CUPOS").
LED Verde		1	Indica que hay cupos disponibles ("CUPOS DISPONIBLES").
Buzzer		1	Emite una alerta sonora cuando se alcanza el aforo máximo.
Resistencias (220 Ω)		3	Limitan la corriente de los LEDs para evitar daños.

Protoboard		1	Facilita el armado del circuito sin necesidad de soldadura.
Cables tipo jumpers		Varios	Conectan los diferentes componentes entre sí y con el Arduino.

3.5. Cálculos

Los cálculos principales del proyecto se centran en la gestión del aforo y los parámetros de control del sistema.

Cálculo de capacidad y umbrales:

- Capacidad máxima del vagón: 6 personas
- Umbral de advertencia (LED amarillo): 3 personas (50% de la capacidad)
- Umbral de capacidad completa (LED rojo + buzzer): 6 personas (100% de la capacidad)

Temporización del sistema:

- Tiempo de anti-rebote para switches: 200 ms
- Duración de la alerta sonora: 2000 ms
- Intervalo de actualización de estado: 500 ms

Lógica de conteo:

- Incremento del contador: activación del switch de entrada
- Decremento del contador: activación del switch de salida
- Validación: el contador nunca puede ser negativo ni superior a la capacidad máxima

Control de indicadores visuales:

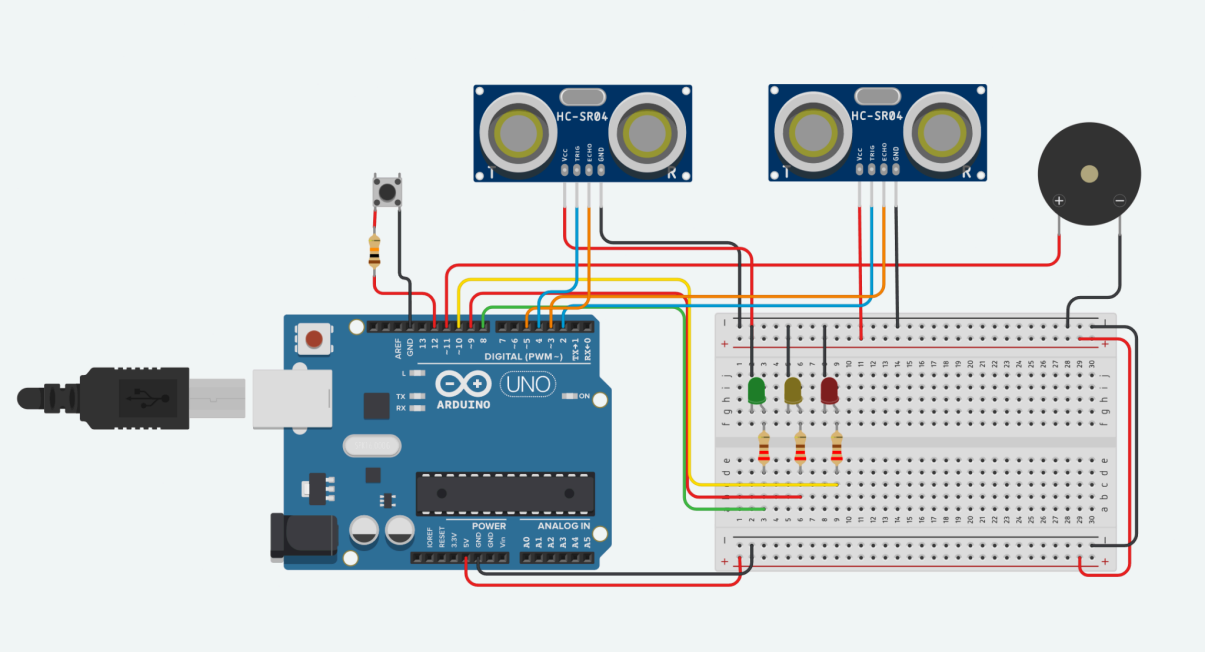
- Estado verde: contador entre 0 y 3 personas
- Estado amarillo: contador entre 4 y 5 personas

ucaldas@ucaldas.edu.co www.ucaldas.edu.co PBX (57)(6) 878 15 00 Calle 65 # 26 - 10 | Manizales - Colombia

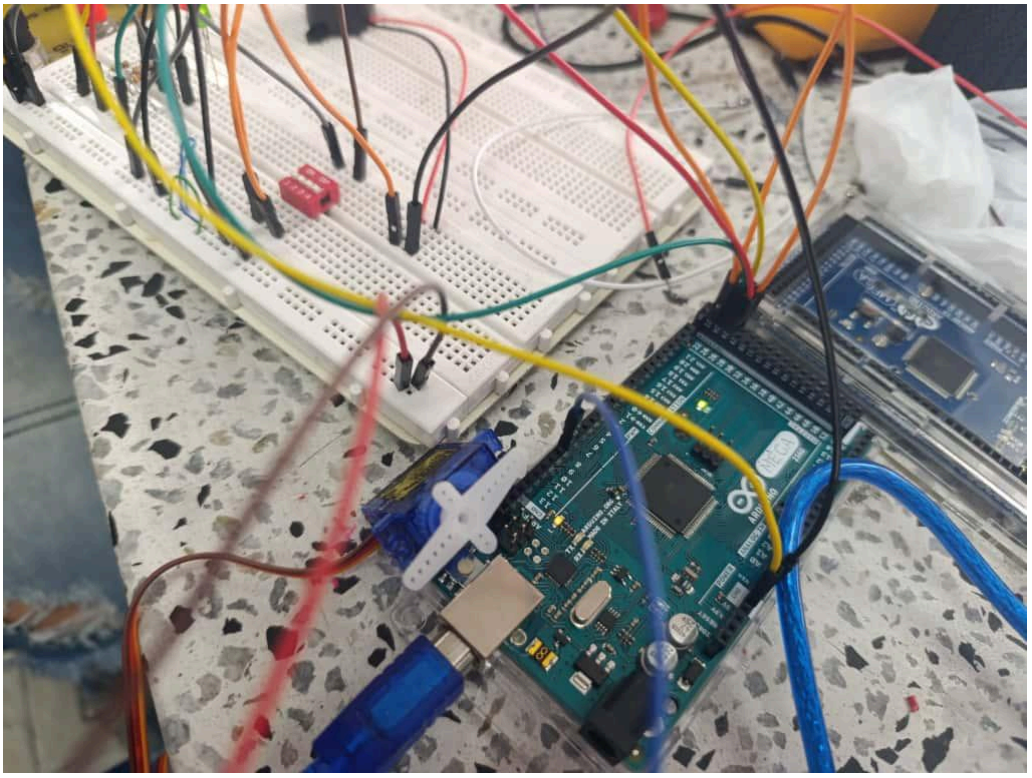
- Estado rojo: contador igual a 6 personas

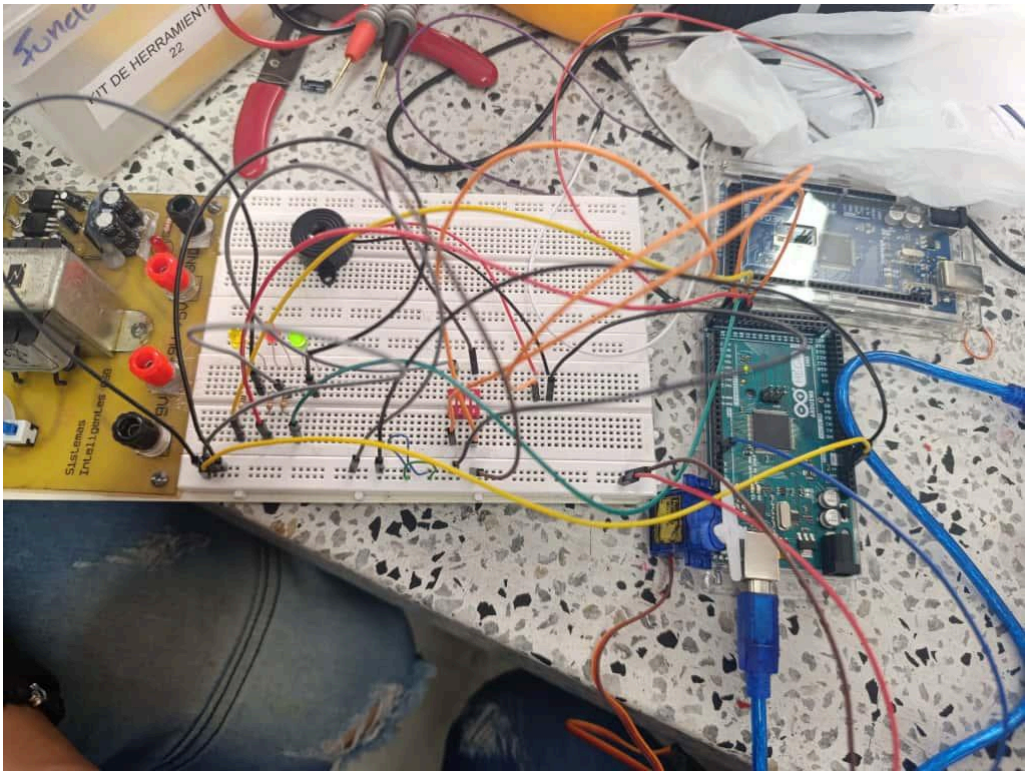
Los cálculos implementados garantizan un control preciso del aforo y una respuesta adecuada de los indicadores según el nivel de ocupación del vagón, alertando cuando se acerca a su capacidad máxima.

3.6. Simulación



3.7. Evidencias





Video de la implementación: CONTROL DE AFORO DEL CABLE AÉREO

4. Código

Código Arduino IDE

```C/C++
// Control de Aforo - Cable Aéreo Línea 3
// Universidad de Caldas

// Definición de pines para switches
#define switchEntrada 2
#define switchSalida 3

// Definición de pines para indicadores
#define ledVerde 8
#define ledAmarillo 9
#define ledRojo 10
#define buzzer 11
```

```
#define botonReset 12

// Variables de control
int aforo = 0;
const int capacidadMaxima = 6;

// Variables para anti-rebote
unsigned long ultimoTiempoEntrada = 0;
unsigned long ultimoTiempoSalida = 0;
const unsigned long delayAntiRebote = 200;

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

  // Configuración de pines
  pinMode(switchEntrada, INPUT_PULLUP);
  pinMode(switchSalida, INPUT_PULLUP);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAmarillo, OUTPUT);
  pinMode(ledRojo, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(botonReset, INPUT_PULLUP);

  Serial.println("Sistema de Control de Aforo Iniciado");
}

void loop() {
  // Lectura de switches con anti-rebote
  if (digitalRead(switchEntrada) == LOW &&
      (millis() - ultimoTiempoEntrada) > delayAntiRebote) {
    if (aforo < capacidadMaxima) {
      aforo++;
      Serial.println("ENTRADA_OK");
    } else {
      Serial.println("ENTRADA_DENEGADA");
    }
    ultimoTiempoEntrada = millis();
  }

  if (digitalRead(switchSalida) == LOW &&
      (millis() - ultimoTiempoSalida) > delayAntiRebote &&
      aforo > 0) {
    aforo--;
    Serial.println("SALIDA_OK");
    ultimoTiempoSalida = millis();
  }

  // Reinicio manual del contador
  if (digitalRead(botonReset) == LOW) {
    aforo = 0;
    Serial.println("RESET_OK");
```


ucaldas@ucaldas.edu.co www.ucaldas.edu.co PBX (57)(6) 878 15 00 Calle 65 # 26 - 10 | Manizales - Colombia

```
    delay(1000);
  }

  // Control de indicadores visuales y sonoros
  actualizarIndicadores();

  // Procesamiento de comandos seriales
  procesarComandosSerial();

  delay(100);
}

void actualizarIndicadores() {
  if (aforo < 3) {
    // Estado verde - Cupos disponibles
    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledAmarillo, LOW);
    digitalWrite(ledRojo, LOW);
    noTone(buzzer);
  } else if (aforo < 6) {
    // Estado amarillo - Últimos cupos
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledAmarillo, HIGH);
    digitalWrite(ledRojo, LOW);
    noTone(buzzer);
  } else {
    // Estado rojo - Vagón lleno
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledAmarillo, LOW);
    digitalWrite(ledRojo, HIGH);
    tone(buzzer, 1000);
  }
}

void procesarComandosSerial() {
  if (Serial.available()) {
    String comando = Serial.readStringUntil('\n');
    comando.trim();

    if (comando == "STATUS") {
      String estado;
      if (aforo < 3) estado = "DISPONIBLE";
      else if (aforo < 6) estado = "LLENANDOSE";
      else estado = "LLENO";

      Serial.print("AFORO:");
      Serial.print(aforo);
      Serial.print(",");
      Serial.print(capacidadMaxima);
      Serial.print(",");
      Serial.println(estado);
```

```
    }
  }
}
```

Código python

Python

```python
import serial
import time
from telegram import Update
from telegram.ext import Application, CommandHandler, ContextTypes
import asyncio
import logging

# Configuración inicial
PUERTO_SERIAL = 'COM14'
BAUDRATE = 9600
TOKEN_BOT = 'TU_TOKEN_AQUI'

# Configuración de logging
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)

arduino = None

def conectar_arduino():
    global arduino
    try:
        arduino = serial.Serial(PUERTO_SERIAL, BAUDRATE, timeout=1)
        time.sleep(2)
        return True
    except Exception as e:
        print(f"Error de conexión: {e}")
        return False

def enviar_comando(comando):
    if arduino and arduino.is_open:
        arduino.write(f"{comando}\n".encode())
        time.sleep(0.5)
```

```
            return leer_respuesta()
        return []

    def leer_respuesta():
        respuestas = []
        while arduino.in_waiting:
            try:
                linea = arduino.readline().decode().strip()
                if linea:
                    respuestas.append(linea)
            except:
                pass
        return respuestas

    def obtener_estado():
        respuestas = enviar_comando("STATUS")
        for resp in respuestas:
            if resp.startswith("AFORO:"):
                try:
                    partes = resp.replace("AFORO:", "").split(",")
                    return int(partes[0]), int(partes[1]), partes[2]
                except:
                    continue
        return None, None, None

    # Handlers de comandos de Telegram
    async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
        mensaje = (
            "🚡 *Sistema de Control de Aforo - Cable Aéreo Línea 3*\n\n"
            "Comandos disponibles:\n"
            "📊 /estado - Estado actual del aforo\n"
            "➡️ /entrada - Registrar entrada\n"
            "⬅️ /salida - Registrar salida\n"
            "🔄 /reset - Reiniciar contador\n"
            "🔧 /diagnostico - Verificar conexión"
        )
        await update.message.reply_text(mensaje, parse_mode='Markdown')

    async def estado(update: Update, context: ContextTypes.DEFAULT_TYPE):
        actual, maximo, estado = obtener_estado()
        if actual is not None:
            if actual < 3:
                emoji = "🟢"
            elif actual < 6:
                emoji = "🟡"
            else:
```


ucaldas@ucaldas.edu.co www.ucaldas.edu.co PBX (57)(6) 878 15 00 Calle 65 # 26 - 10 | Manizales - Colombia

```
            emoji = "🔴"

        mensaje = (
            f"{emoji} *Estado del Aforo*\n\n"
            f"👥 Personas: {actual}/{maximo}\n"
            f"📍 Estado: {estado}"
        )
    else:
        mensaje = "❌ Error de comunicación con Arduino"

    await update.message.reply_text(mensaje, parse_mode='Markdown')

# [Resto de handlers...]

def main():
    if not conectar_arduino():
        return

    application = Application.builder().token(TOKEN_BOT).build()

    # Registro de comandos
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("estado", estado))
    application.add_handler(CommandHandler("entrada", entrada))
    application.add_handler(CommandHandler("salida", salida))
    application.add_handler(CommandHandler("reset", reset))
    application.add_handler(CommandHandler("diagnostico",
diagnostico))
    application.add_handler(CommandHandler("ayuda", ayuda))

    application.run_polling()

if __name__ == '__main__':
    main()
```

5. Resultados obtenidos

Control de acceso automático:

El servomotor que simula la barrera respondió de forma precisa a los eventos de entrada y salida. Al detectar un vehículo en el sensor de entrada, la barrera se

abría automáticamente siempre que hubiera cupos disponibles, cerrándose luego de permitir el acceso.

Gestión de cupos en tiempo real:

El sistema mantuvo un conteo dinámico de los espacios disponibles. El valor se actualizó correctamente tanto en las entradas como en las salidas, evitando desbordamientos (no permitía ingresar más de la capacidad máxima).

Indicadores luminosos efectivos:

El LED verde se activaba cuando había disponibilidad, mientras que el LED rojo se encendía al llenarse el parqueadero, funcionando como señalización clara para los usuarios.

Estabilidad del sistema:

Durante las pruebas de laboratorio, el sistema se mantuvo estable con múltiples simulaciones de ingreso y salida, mostrando confiabilidad en la actualización de datos y control de actuadores.

6. Conclusiones

Aprendizaje del proyecto:

El desarrollo de este sistema permitió comprender la integración de componentes hardware y software en una solución IoT completa. Se consolidaron conocimientos en programación de Arduino, manejo de comunicación serial, desarrollo de bots en Telegram y creación de interfaces entre sistemas embebidos y aplicaciones web. La implementación del anti-rebote para switches demostró la importancia de considerar aspectos físicos en la lectura de sensores.

Relevancia para el mundo real:

Este prototipo evidencia el potencial de las tecnologías IoT para optimizar sistemas de transporte masivo. La capacidad de monitorear el aforo en tiempo real y notificar a los operadores mediante Telegram representa una solución escalable que podría implementarse en el cable aéreo real, mejorando la seguridad, eficiencia y experiencia del usuario en el servicio de transporte.

Dificultades encontradas:

La principal dificultad fue la adaptación del proyecto al usar switches en lugar de sensores ultrasónicos, lo que requirió ajustar la lógica de detección y implementar mecanismos de anti-rebote. También se presentaron desafíos en la estabilidad de la comunicación serial entre Arduino y Python, especialmente en la sincronización de comandos y respuestas, que se resolvió mediante reintentos automáticos y manejo de excepciones.

Logros alcanzados:

Se logró desarrollar un sistema funcional que integra exitosamente todos los componentes: detección mediante switches, control de indicadores visuales y sonoros, comunicación serial estable y un bot de Telegram completamente operativo. El sistema demostró confiabilidad en el conteo de personas, respuesta adecuada de los indicadores según los umbrales definidos, y capacidad de monitoreo remoto en tiempo real, cumpliendo con todos los objetivos planteados inicialmente.

7. Referencias

Arduino. (s.f.). Arduino language reference. https://www.arduino.cc/reference/en/
Python Software Foundation. (s9f.). Python documentation. https://docs.python.org/3/
Telegram. (s.f.). Bot API documentation. https://core.telegram.org/bots/api
Python-Telegram-Bot. (s.f.). python-telegram-bot documentation. https://python-telegram-bot.org/
PySerial. (s.f.). Serial communication in Python. https://pyserial.readthedocs.io/
Microchip Technology Inc. (2018). ATmega328P datasheet. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf
Tinkercad. (s.f.). Tinkercad Circuits. https://www.tinkercad.com/circuits
Arduino. (s.f.). Serial communication. https://www.arduino.cc/reference/en/language/functions/communication/serial/
GitHub. (s.f.). python-telegram-bot examples. https://github.com/python-telegram-bot/python-telegram-bot/tree/master/examples
Stack Overflow. (s.f.). Serial communication troubleshooting. https://stackoverflow.com/