UNIVERSIDAD VERACRUZANA
FACULTAD DE INSTRUMENTACIÓN ELECTRONICA
XALAPA, VER

Experiencia Educativa

Programación Avanzada

Docente

Domínguez Chávez José Alfonso

Practica

Integración de sensores y procesamiento de datos con Arduino

Integrantes

Hernández Pérez Luis Gerardo

Sánchez Hernández Christian

Brígido Mora Saín David

Roa Herrera Oswaldo

Viernes 28 de marzo de 2025

Introducción:

En este reporte de trabajo observaremos como es que trabajan los programas de Arduino y Spyder para así poder tener una buena integración con algunas variables de salida (sensores).

La finalidad de esta práctica es la implementación de un flujo de datos robusto y confiable.

Materiales:

Sensor ultrasónico HC-SR04

Sensor PIR AM312

Lector de tarjetas RFID RC522

Fotorresistor

Resistencias de 5 k

Protoboard

Jumpers

Arduino

Implementación de códigos:

```python
import serial
import pandas as pd
from datetime import datetime
import time

try:
    arduino = serial.Serial('COM3', 9600, timeout=1)
    time.sleep(2)
    datos = []

    print("Sistema listo. Monitoreando sensores...")

    while True:
        if arduino.in_waiting > 0:
            linea = arduino.readline().decode('utf-8').strip()

            if linea and all(key in linea for key in ["RFID:", "PIR:", "US:", "LDR:"]):
                registro = {'Timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S')}
                for parte in linea.split(','):
                    clave, valor = parte.split(':', 1)
                    registro[clave] = valor

                datos.append(registro)
                print("Datos:", registro)

                if len(datos) % 10 == 0:
                    pd.DataFrame(datos).to_csv('datos_sensores.csv', index=False)

except KeyboardInterrupt:
    arduino.close()
    pd.DataFrame(datos).to_csv('datos_sensores.csv', index=False)
    print("Datos guardados exitosamente.")
```

Este código establece comunicación con un Arduino para recibir, procesar y almacenar datos de sensores. Aquí su funcionamiento esencial:

1. **Configuración inicial**:
    - Abre comunicación serial con Arduino (puerto COM3 a 9600 bps)
    - Prepara lista vacía datos para almacenar registros
    - Espera 2 segundos para estabilizar la conexión
2. **Procesamiento continuo**:
    - Monitorea constantemente el puerto serial
    - Cuando detecta datos:

- Decodifica la línea recibida (formato UTF-8)
- Verifica que contenga todos los sensores esperados (RFID, PIR, US, LDR)
- Añade marca de tiempo precisa al registro

3. **Estructuración de datos**:
    - Divide la cadena recibida (ej: "RFID:ABC,PIR:1,US:30,LDR:80")
    - Convierte cada valor en entradas de diccionario
    - Agrega el registro completo a la lista datos
4. **Almacenamiento**:
    - Muestra cada registro en consola para monitoreo en tiempo real
    - Cada 10 registros, guarda automáticamente en CSV (datos_sensores.csv)
    - Al interrumpir (Ctrl+C), guarda todos los datos pendientes y cierra la conexión

Formato de salida:
Cada línea en el CSV contiene:

- Timestamp (fecha y hora exacta)
- Valores de los 4 sensores
- Ejemplo: 2023-11-15 14:30:00, RFID:ABC123, PIR:1, US:25, LDR:65

Código en Arduino:

```
#include <SPI.h>   // Se llama la libreria para la comunicación SPI
#include <MFRC522.h>   //Se llama la librería para controlar el módulo lector de tarjetas RFID MFRC522
// A continuación se definen los pines que ocupamos en el arduino para los sensores
#define RST_PIN 9
#define SS_PIN 10
#define PIR_PIN 2
#define TRIG_PIN 5
#define ECHO_PIN 6
#define LDR_PIN A0
// Creación del objeto mfrc522 para controlaar el lector RFID
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup() {
  Serial.begin(9600);  // Se inicia la comunicación serial
  SPI.begin();   // Inicia comunicación SPI
  mfrc522.PCD_Init();   // Inicialización del lector RFID
  pinMode(PIR_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  // PIR con filtro de ruido
  int movimiento = 0;
  for(int i=0; i<10; i++) {
    movimiento += digitalRead(PIR_PIN);
    delay(50);
  }
  movimiento = (movimiento > 5) ? 1 : 0;

  // Ultrasónico
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int distancia = pulseIn(ECHO_PIN, HIGH) * 0.034 / 2;

  // LDR calibrado
  int raw_ldr = analogRead(LDR_PIN);
  int luz = map(raw_ldr, 200, 800, 100, 0); // Ajusta estos valores

  // RFID
  String tarjeta = "N/A";
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      tarjeta += String(mfrc522.uid.uidByte[i], HEX);
    }
    mfrc522.PICC_HaltA();
  }

  // Enviar datos por serial
  Serial.print("RFID:" + tarjeta +
               ",PIR:" + String(movimiento) +
               ",US:" + String(distancia) +
               ",LDR:" + String(luz));
  Serial.println();

  delay(300); // Se le agrega un retardo de 300ms antes de repertir el loop, para no saturar el serial.
}
```

Este código lee 4 sensores y envía sus datos por comunicación serial. Aquí su estructura esencial:

1. **Configuración inicial**:
    - Inicializa comunicación serial (9600 bps) y protocolo SPI (para RFID)
    - Define pines para cada sensor:
        - RFID (pines 9 y 10)
        - PIR (pin 2 - movimiento)
        - Ultrasónico (pines 5 y 6 - distancia)
        - LDR (pin A0 - luz ambiental)
2. **Lógica principal (loop)**:

a) Sensor PIR (movimiento):

- Toma 10 muestras con 50ms de intervalo
- Considera "movimiento detectado" (valor 1) si >5 muestras son positivas
- *Objetivo*: Filtra falsos positivos

b) Sensor ultrasónico:

- Envía pulso de 10µs por el pin TRIG
- Mide tiempo de retorno del eco (pin ECHO)
- Calcula distancia en cm: (tiempo × 0.034) / 2

c) Sensor Fotorresistor (luz):

- Lee valor analógico (0-1023)

- Convierte a porcentaje (100%-0%) con ajuste personalizable
- *Ejemplo*: map(valor, 200, 800, 100, 0)

d) Lector RFID:

- Detecta tarjetas presentes
- Lee UID y lo convierte a hexadecimal
- Devuelve "N/A" si no hay tarjeta

3. **Envío de datos**:
   - Formato uniforme: "RFID:valor,PIR:valor,US:valor,LDR:valor"
   - Ejemplo real: "RFID:5A3BC1,PIR:1,US:37,LDR:80"
   - Incluye salto de línea (println()) para cada lectura
4. **Temporización**:
   - Delay de 300ms entre ciclos
   - *Propósito*: Evitar saturación del puerto serial

Características clave:

- Filtrado inteligente para PIR
- Cálculo preciso de distancia
- Adaptable a diferentes condiciones de luz (LDR)
- Compatible con múltiples tarjetas RFID
- Estructura de datos clara para fácil procesamiento

Salida típica:

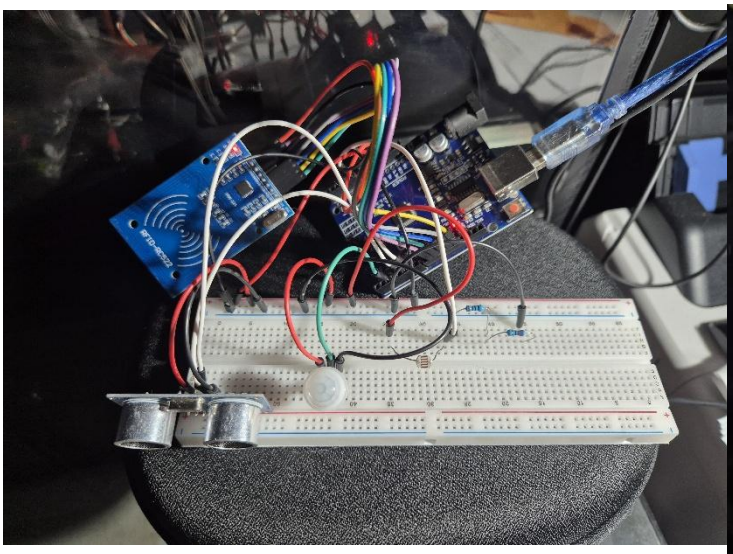
Copy

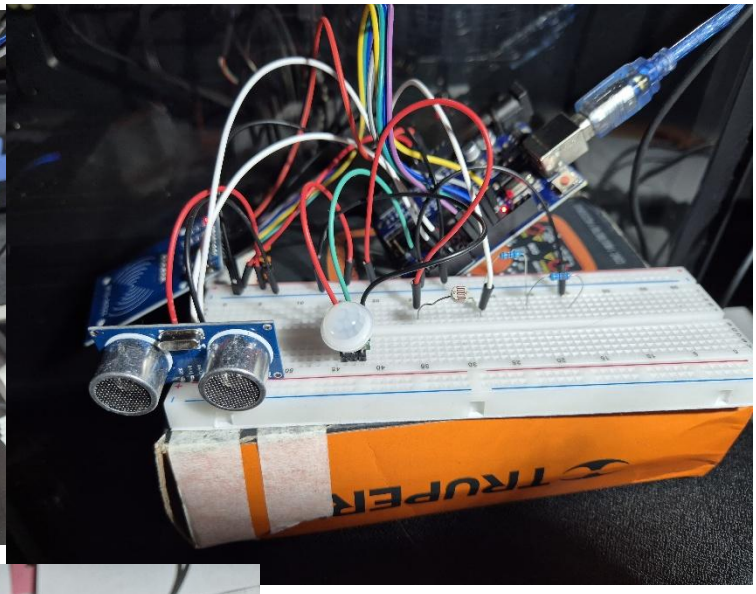
RFID:N/A,PIR:0,US:255,LDR:30

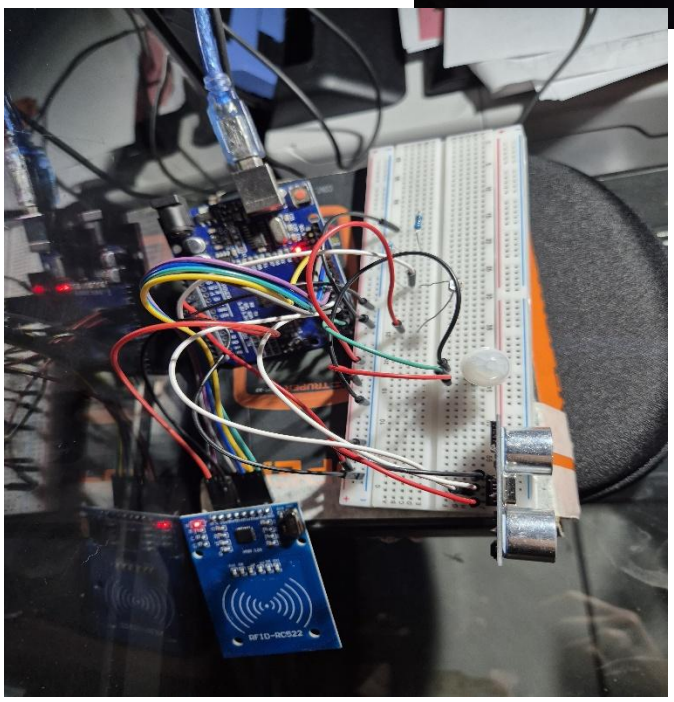
RFID:AB3C41,PIR:1,US:120,LDR:15

Este código se complementa perfectamente con el script Python previamente explicado, formando un sistema completo de monitoreo.

Ensamblado:







Nota*

En la fotorresistencia su valor aumenta mientras no exista ningún tipo de luz (LDR), para este se ocuparon dos resistencias de 5K en serie, (ya que se necesitaba una de 10k pero al no haberla se optó por usarlas en serie)

El sensor ultrasónico mientras el objeto más se acerca más se hace notar en su sistema y viceversa

El sensor PIR es el detector de movimiento, pone 1 cuando se detecta algo 0 cuando no se detecta nada

El sensor RFID se pone N/A cuando se detecta la tarjeta (se han colocado 3 diferentes tarjetas solo que no se pueden apreciar bien)

Se puso un delay de 300 para que sea más fácil que PIR y la foto resistencia puedan leer correctamente los valores, ya que si este se aumentaba los datos salían erróneos.