Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

**Tampico, Tamaulipas a 08 de noviembre de 2022**

***Mando para Videojuego***

***Proyecto de Integrador***

**Integrantes de Equipo:**

**Arellano Trapala Juan Eduardo**

**Dávila Vitales Luis Ángel**

**Moreno Galicia Brandon Antonio**

**Suárez Martínez Julio César**

**Profesor:** **Dr. García Ruiz Alejandro Humberto**

**Asignatura:** **Diseño Electrónico Basado en Sistemas Embebidos**

**8vo. Semestre – Grupo “I”**

**2022-3**

### 

### *Mando de videojuego*

**Descripción:** Programación y diseño de un mando para videojuegos basado en microcontrolador.

**Introducción:**

Para el funcionamiento lógico del dispositivo se ha utilizado una placa de programación Arduino UNO R3 por lo que la codificación del programa cargado al microcontrolador se ha realizado en el IDE propio de la marca. Adicionalmente se ha optado por complementar la programación con los módulos para Python: Pyserial, Pynput y Pyqt5.

Para la programación del dispositivo se han utilizado las siguientes instrucciones:

**Arduino IDE.**

**Serial.begin()**

Define el rango de datos en bits por minuto (baudios) para la transmisión serial de datos.

**pinMode()**

Establece el modo en que funcionará el pin especificado, entrada (INPUT) o salida (OUTPUT).

**digitalRead()**

Lee el valor digital del pin especificado, LOW o HIGH

**millis()**

Retorna el número de milisegundos transcurridos desde que la placa Arduino comenzó a correr el programa actual.

**Serial.println()**

Imprime datos al puerto serial como texto en código ASCII seguido de un retorno de carro (\r) y nueva línea (\n).

**Python**

**Pyqt5**

**timeout**

Establece el tiempo en que será ejecutada una función continuamente.

**Pynput**

**Controller.press()**

Digita el carácter (tecla) especificada como argumento.

**Controller.release()**

Libera la tecla presionada anteriormente con la función press().

**Pyserial**

**serial.Serial()**

Establece la conexión con la tarjeta Arduino especificando el puerto y la velocidad del puerto serial. Es una instancia por lo que debe ser asignado a una variable.

**serial.open()**

Inicia la comunicación serial desde y hacia el puerto de la tarjeta.

**serial.close()**

Termina la comunicación serial con la tarjeta Arduino.

**serial.readline()**

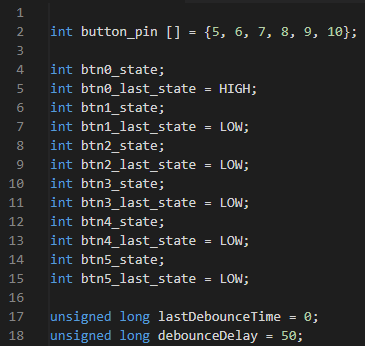
Lee una cadena enviada por la tarjeta Arduino.

**Desarrollo:**

La elaboración de este proyecto puede dividirse en dos partes, por un lado, la programación y por el otro el diseño electrónico del dispositivo.

**Arduino**

En la línea 1 son declarados dentro de un arreglo de tipo entero los pines utilizados para la conexión de los botones.

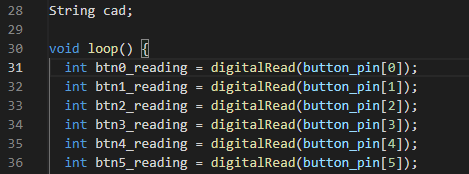


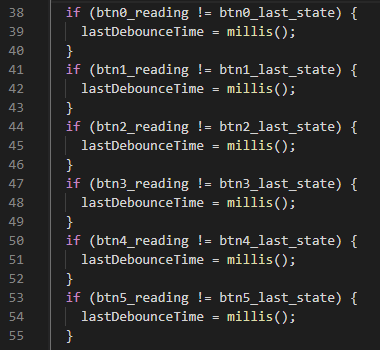
De la línea 4 a 15 se declaran variables del tipo entero que tomaran el valor HIGH o LOW (1 o 0). Estas representan el estado actual y el ultimo estado del botón.

Las constantes declaradas en las líneas 17 y 18 servirán para verificar el tiempo de retardo entre una pulsación y otra. Esto se realiza debido a que los botones por sus características pueden llegar a registrar pulsaciones (rebotes) que no son realizadas por el usuario, ya que estas pulsaciones suelen ser muy rápidas (en el rango de milisegundos) podemos establecer un tiempo (debounceDelay) para evaluar si una pulsación fue realizada por el usuario o por un rebote del botón.

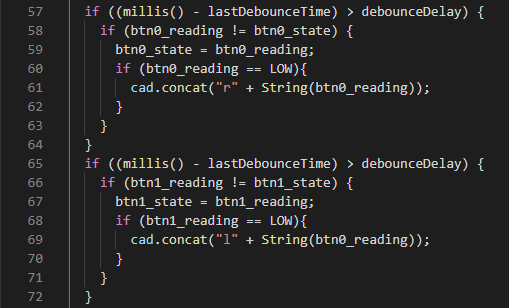
Texto

Descripción generada automáticamenteEn el método setup() se establece la velocidad de comunicación. Así mismo el modo de operación de los pines correspondientes a los botones, es decir, como pines de entrada (INPUT). Para realizar esto de forma eficiente se utiliza una estructura iterativa.

****Al iniciar el método loop() se lee el estado de cada botón a su variable correspondiente mediante la función digitalRead().



A continuación, se verifica si el ultimo estado del botón ha cambiado, si es así se asigna el valor retornado por la función millis() a la variable lastDebounceTime.

****Una vez obtenido el tiempo en que se presionó por última vez el botón, evaluaremos si esto se ha debido al usuario o a un rebote. Para esto restaremos el tiempo actual menos el milisegundo en que fue presionado. Si la diferencia es mayor que debounceDelay, es decir el máximo tiempo en que se puede considerar un rebote del botón, quiere decir que el botón fue presionado por el usuario. Entonces ahora se evalúa si la lectura es diferente al estado del botón y si es así entonces el estado del botón será igual a la lectura. La última evaluación es la que nos dará el valor que esperamos para saber si el botón se encuentra presionado y mandarlo a Python mediante una cadena.

Esta operación se realiza para cada botón, por lo que, si más de uno es presionado al mismo tiempo, serás concatenados a la cadena que será enviada. Para saber qué valor corresponde a cierto botón se agregará antes de este un carácter identificador, por ejemplo “r” y “l” para los botones que realizan el movimiento de un personaje en estas direcciones.

**QtDesigner**

Para llevar a cabo la ejecución de la recepción y procesamiento de los datos obtenidos por el microcontrolador a través de los botones fue necesario utilizar la clase Time del módulo Pyqt5, por lo que se diseño una interfaz simple con el software QtDeigner.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

le\_com

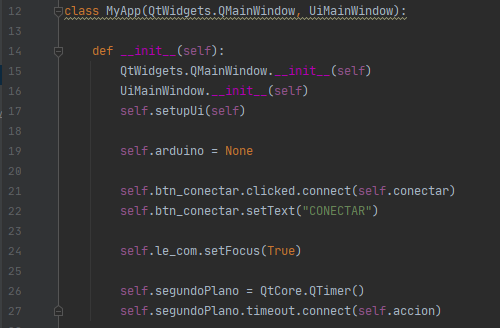
btn\_coenctar

lw\_datos

Como se puede apreciar la interfaz cuenta solo con lo necesario para establecer la comunicación con el microcontrolador ingresando el número de puerto en le\_com y monitorear los datos leídos que se mostraran en lw\_datos.

**Python**

Para la ejecución de la interfaz y eventualmente la lectura de datos del microcontrolador se creó la clase necesaria para inicializar los componentes y vincular las funciones que se ejecutaran.

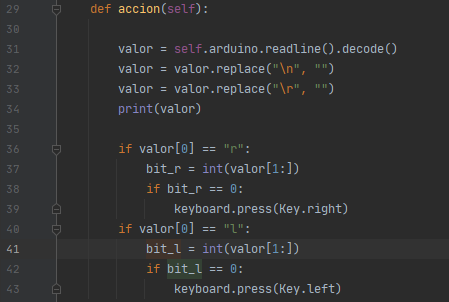


Declaración del constructor de la interfaz, así como la vinculación de las funciones con sus respectivos componentes como lo es en las líneas 21 y 22 para el botón que establecerá la conexión con el microcontrolador. En la línea 26 se crea la instancia de la clase timar y en la siguiente línea se vincula a la función acción.



En la función conectar las principales instrucciones son la instanciación de la clase Serial en la línea 52 y el inicio de ejecución de la función vinculada al QTimer en las líneas 54 y 59.

De forma breve este segmento verifica si la instancia Serial existe, si no es así la crea, si existe abre la conexión a ella y si el botón en la interfaz es presionado para desconectarle la ejecución del QTimer se detiene.

En la función acción de las líneas 31 a 33 leemos y decodificamos la cadena generada por el Arduino, así mismo eliminamos caracteres que pudiesen generar ruido al momento de extraer los datos necesarios como el salto de línea y el retorno de carro.

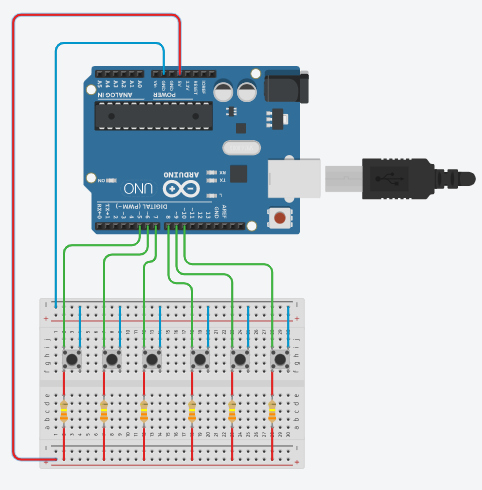
Una vez que tenemos nuestra cadena “limpia” procedemos a verificar los caracteres que necesitamos y que realizar basándonos en los identificadores que se agregaron en Arduino.

**Diseño Electrónico**

**Los componentes físicos utilizados para este proyecto fueron:**

* **Arduino UNO R3**
* **Tabla de prototipos**
* **6 push button**
* **6 resistencias 330kΩ**

**Las conexiones se muestran en el siguiente esquema.**



**Diagrama, Esquemático

Descripción generada automáticamente**