**Universidad Autónoma de Tamaulipas**

Facultad de ingeniería Arturo Narro Siller

Materia: Diseño Electrónico Basado en Sistemas Embebidos

Nombre: Gonzales Saldívar Luis Roberto

Guerrero Gamez Francisco Javier

Martínez Reyes Fernando

Sánchez Ramírez Alan Ariel

Villalobos de León Juan Carlos

Grupo: 8-I

Maestro: García Ruiz Alejandro Humberto

Documentación del Proyecto integrador: Control de videojuegos basado en Arduino y Python.

**Introducción**

Crear un sistema embebido para la simulación de un control de videojuegos utilizando los conocimientos aprendidos en la clase y los recursos disponibles y accesibles a nuestro alcance, con la finalidad que pueda funcionar con la ejecución de un emulador

**Planeación y Prototipo inicial**

El primer paso para la realización de este proyecto fue, la planeación que componentes e ideas pudiéramos usar para la realización de este proyecto, desde la utilización de carcasas de controles, cartón o incluso armar un modelo de control básico con tablas de madera, todo esto como una base para nuestro propio control.

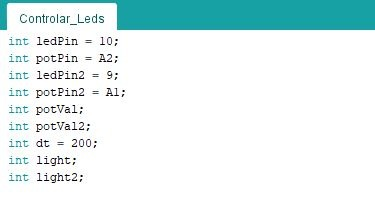
La utilización de una placa Arduino UNO, protoboard, botones, resistencias de 330 kiloohms, Cables Dupont para Protoboard M/H, leds

Una vez teniendo en mente como armaremos desde la carcaza del control hasta los materiales que utilizamos, procedimos a armar las primeras ideas para un control firme y funcional.

Cabe mencionar que para la realización de este proyecto en Arduino, se tuvieron que realizar 2 versiones para demostrar los diferentes avances que se realizaron durante su desarrollo.

**Programación.**

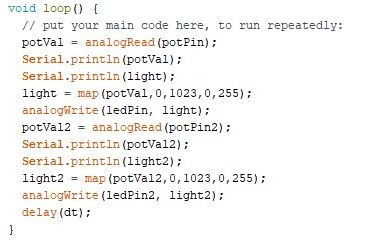
**versión 1**



1.- En este segmento de código en Arduino, se asignan los pines análogos que va a pertenecer cada botón o pulsador y cada led.



2.- En esta parte del setup se configura el pinMode en el caso del “led” se maneja como entrada y en el caso de pulsador se maneja como salida. Utilizando la unidad de medida baudios en 9600.



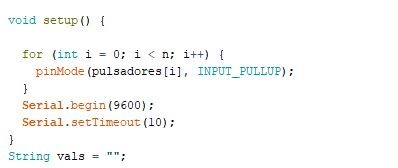


3.- En el void loop, se lee el valor del pulsador, para posteriormente imprimirlo en el Serial, de la misma forma, lo hace con el pulsador numero 2, que a su vez asignándole la variable ligth y ligth2 el promedio de cada potenciómetro, dividiendo su valor digital sobre el análogo (1023/255).

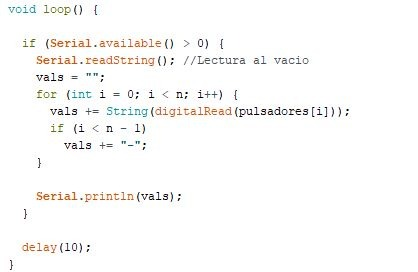
**VERSIÓN 2**



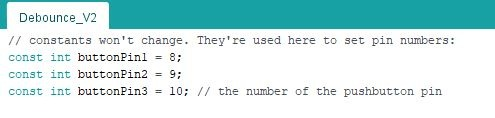
4.- Posteriormente se declara la variable N con un valor de 3 con el tipo de dato int y un arreglo llamado pulsador con los valores 8,9,10.



4.- Una vez en el setup, se hace el uso de un ciclo for para recorrer todos los pulsadores y usando la resistencia interna de “INPUT\_PULLUP” para detectar la pulsación de un botón así mismo, usamos nuevamente la unidad de medida de baudios(9600) con un SetTimeOut de 10 milisegundos. Por ultimo se declara una variable llamada “vals” el cual tiene un valor vacío.



5.- Una vez dentro del loop, se hace el uso de una condicional if, para comprobar si el serial esta libre, se hace una lectura de vacio, esto para evitar la sobresaturación de datos al momento de enviar las lecturas de cada pulsación. Luego de esto se recorre el arreglo con el uso de un ciclo hasta “N” números y se imprime el valor de la variable vals.



6.- Necesitamos usar Debounce para corregir un problema que tenia el programa anterior, tenia mucho delay, debido al exceso de datos obtenido atraves del Arduino y el uso del segundo plano. Se declaran 3 variables const de tipo int llamadas buttonPin1,2,3 además de siguir el numero al pin que está conectado en el Arduino en este caso es 8,9 y 10

Tabla

Descripción generada automáticamente con confianza media

7.- Posteriormente se declaran 3 variables de tipo long y se le asigna valor 0. Además de otra variable long llamada debounceDelay que se le asigna el valor de 50, además de declarar e instanciar la variable n de tipo int con valor 3, y se declara un arreglo de llamado pulsadores con los números 2,9,10

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

8.- Se le aplica una resistencia pullup a las 3 entradas digitales y con pin mode configuramos el modo de trabajo de los pones, después en un for de i hasta n, se le aplica lo dicho anteriormente pero de cada elemento del arreglo, es decir a los pines 8,9 y 10

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

9.- Con digital read ponemos el valor de buttonpin1 en la variable de tipo int Reading. Después en un if si Reading es diferente o igual a lastbuttonState1. Se obtendrá la cantidad de milisegundos que ha pasado desde que comenzó la ejecución del programa y se signará lastDebounceTime1

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

10.- Con digital read ponemos el valor de buttonpin2 en la variable de tipo int Reading. Después en un if si Reading es diferente o igual a lastbuttonState2. Se obtendrá la cantidad de milisegundos que ha pasado desde que comenzó la ejecución del programa y se signará lastDebounceTime2



Interfaz Qt Designer

11.- Es una interfaz simple, cuenta con un botón conectar, y cuenta con line edit para poner el número de puerto al que está conectado el Arduino, y por último un list widget en la cual se visualizaran los datos obtenidos desde Arduino.

Texto

Descripción generada automáticamente

12.- En estas líneas de código se realizan las importaciones de diversos módulos que se necesitaran para que el programa funcione de manera correcta, entre estas importaciones se encuentra el serial para conectar con Arduino, keyboard , key y controller.

Interfaz de usuario gráfica

Descripción generada automáticamente

13.- Se declara el arreglo vacío pulsadores, además de declarar e instanciar la variable puls con valor de 1.

Texto

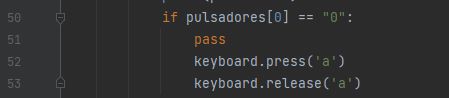
Descripción generada automáticamente

14.- En estas líneas de código se conecta el btn\_conexion con un método dentro del programa. Además de establecer el segundo plano con QtCore.QTimer, y asignarle igualmente un método.

Texto

Descripción generada automáticamente

15.- En el método acción con parámetro self, en este método se guarda la línea de Arduino en la variable local v de pyhton, y con print se imprime el valor en consola, además de asignarle a la variable valor el contenido de v acomodado y separándolo por “-“, al final imprimiendo la variable pulsadores



16.- A continuación se usa una condicional, en el cual si en el arreglo pulsadores en la posición 0 es igual a 0, significa que se presiona la letra A, y se utiliza la función Keyboard.press para soltar dicha tecla y en esa forma se verá el valor de ella.

Texto

Descripción generada automáticamente con confianza baja

17.- A continuación, se usa una condicional, en el cual si en el arreglo pulsadores en la posición 1 es igual a 0, significa que se presiona la letra B, y se utiliza la función Keyboard.press para soltar dicha tecla y en esa forma se vera el valor de ella.

Texto

Descripción generada automáticamente con confianza baja

18.- A continuación se usa una condicional, en el cual si en el arreglo pulsadores en la posición 2 es igual a 0, significa que se presiona la letra C, y se utiliza la función Keyboard.press para soltar dicha tecla y en esa forma se vera el valor de ella

Texto

Descripción generada automáticamente

19.- Se crea el método conexión el cual esta vinculado al botón conexión y en la variable “V”. En el método conexión se obtiene el texto del btn\_conexion y se guarda en la variable local v, posteriormente con un if se compara si el texto del botón es igual a “conectar” se cambia el texto del botón a desconectar.

Texto

Descripción generada automáticamente

20.- Se usa el uso de la condicional if para comparar si Arduino es igual a “none” si nunca se realizo la conexión, y posteriormente procede a realizarla

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

21.- Luego se utiliza la condicional “elif not” para establecer la conexión y restablecerla, además haciendo uso del segundo plano, con el valor indicado para que se ejecute cada 5 milisegundos.

Anteriormente se había logrado dicha conexión y la transmisión de datos pero habiendo un problema, el cual es, que después de cierto tiempo (10 segundos de ejecución) se dejaba de transmitir información, esto debido al número de datos abismales.

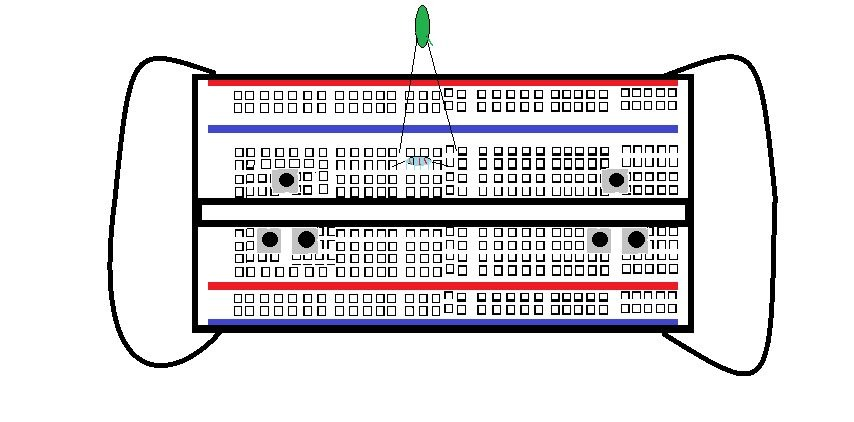
Encontrando la solución a esto, se utilizó la función del segundo plano por el cual se procedió a integrar una interfaz gráfica, ya que QtDesigner cuenta con esta función (segundo plano)

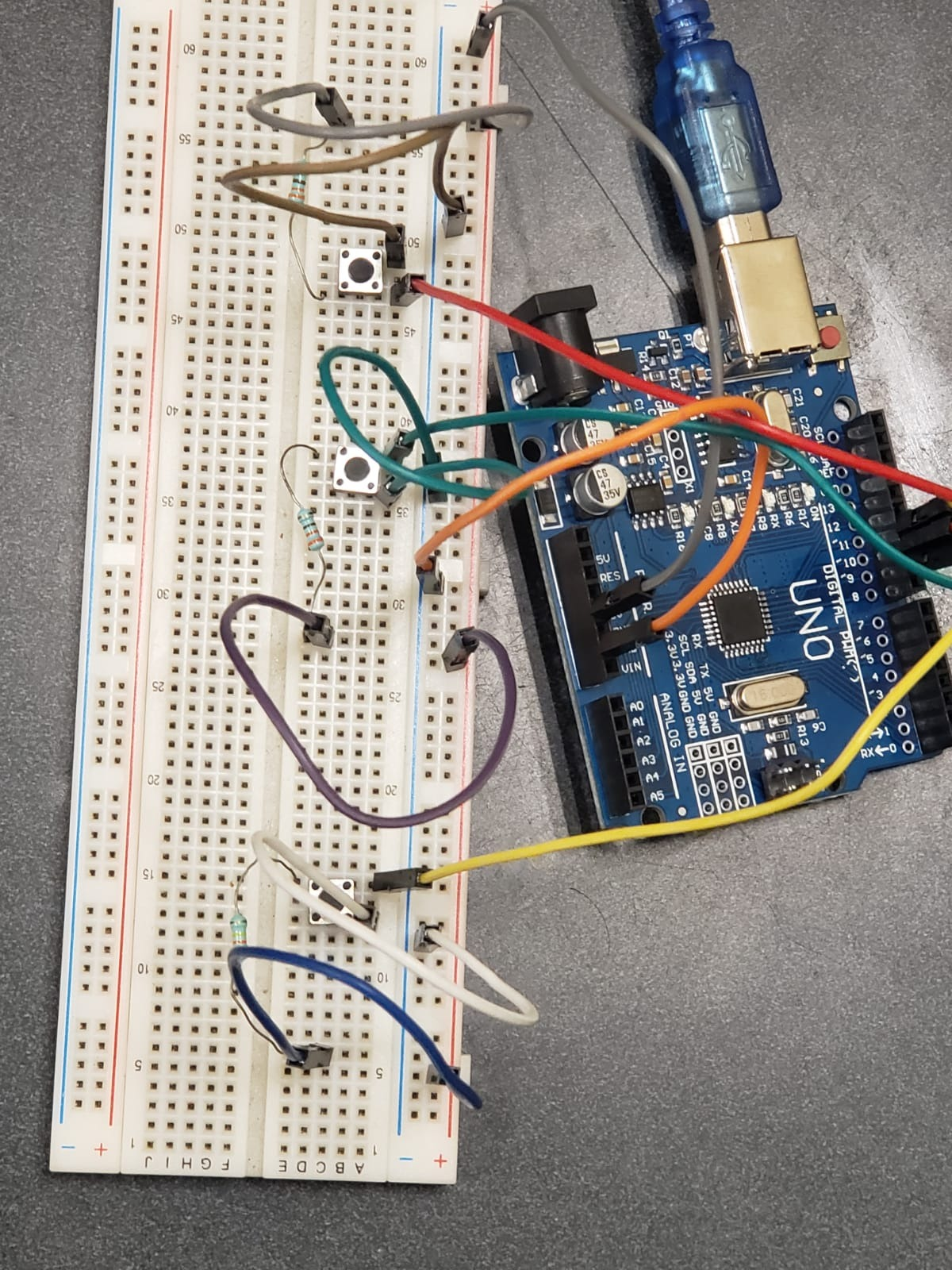
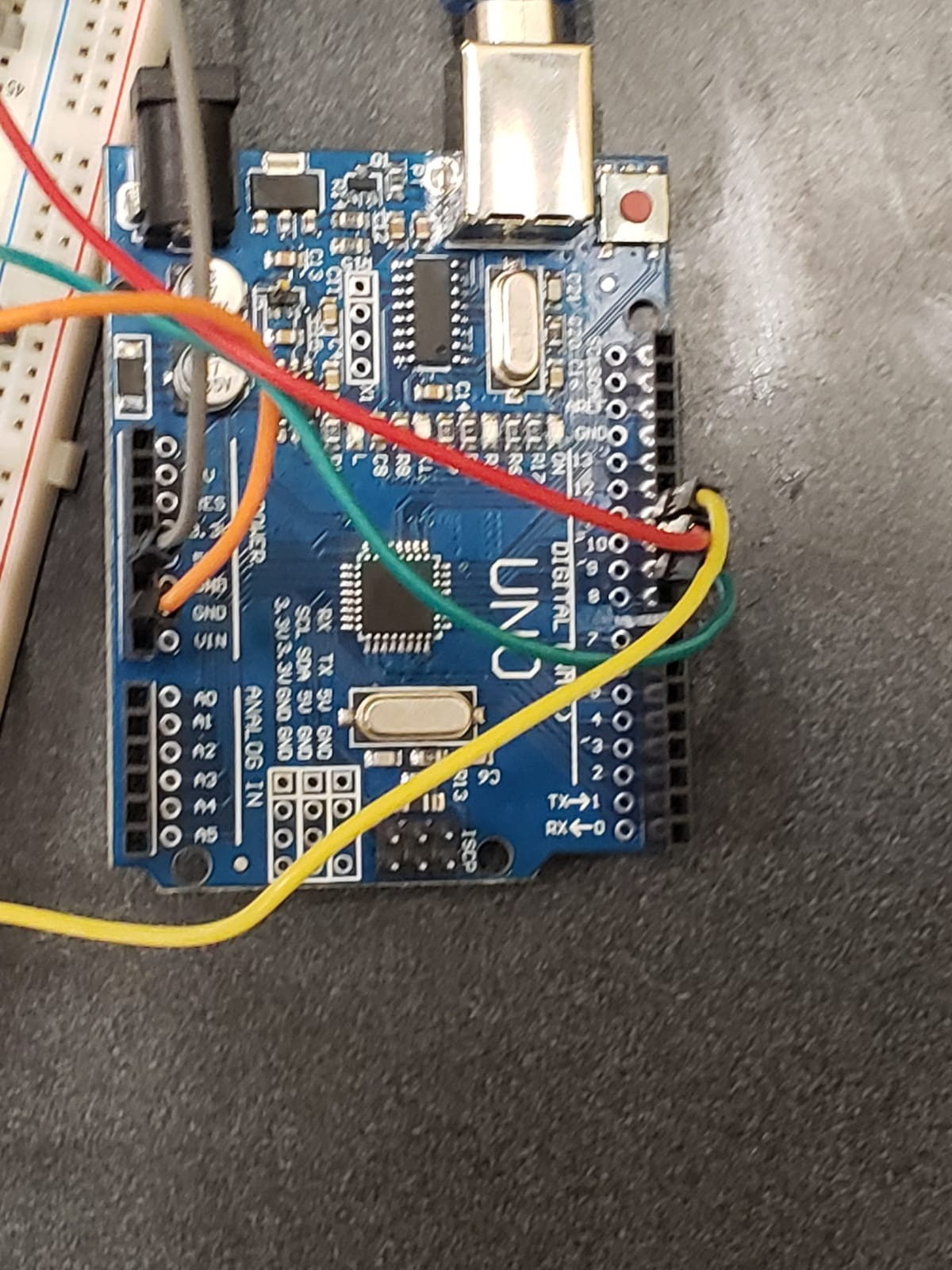
Texto

Descripción generada automáticamente

22.- Al final de este código se implementa una condicional “else”, el cual pasa del estado conectado al desconectado.

Boceto de control inicial Boceto de control Final



Proceso de armado de control

Corridas de prueba