PRODUCTO DE UNIDAD N.3

Jhonatan Tituaña, Bryan Azuero, Javier Arteaga

Resumen -

En el presente artículo se muestran los resultados de la práctica de la asignatura Arquitectura de computadoras, en la cual se busca obtener el conocimiento acerca del uso del simulador withcode. Diseñaremos dos ejercicios de programación elaborada con el lenguaje de programación Python empleando los conceptos de programación orientada a objetos, cada función tanto de la alarma y el sistema de riego estará enlazada con un pin virtual de la Raspberry PI y será puesta a prueba con la ayuda del simulador with code.

Objetivo General

Desarrollar una alarma de incendios y un sistema de riego utilizando el lenguaje de programación Python y empleando los conceptos de programación orientada a objetos, qué seleccione las operaciones matemáticas por medio de la lectura de los puertos virtuales de una Raspberry PI.

Índice de Términos –

Python, POO, Raspberry PI, simulador, GPIO.

I. INTRODUCCION

En el presente artículo analizaremos el problema propuesto en el producto de unidad, la elaboración de una alarma de incendios y un sistema de riego automático aplicando el lenguaje de programación Python tiene muchas ventajas para enseñar el primer curso de programación. Su sintaxis sencilla y su modo interactivo hace que éste sea ideal para la enseñanza. Es importante que es la programación orientada a objetos, conocer el paradigma de programación que más se utiliza en la actualidad. Esta práctica está destinada a explicar el desarrollo de la práctica de programación ocupando funciones que nos proporcionar la librería especializadas como son la de los puertos virtuales GPIO, y la librería time. Para ejecutar nuestro programa necesitaremos la ayudad de un si simulador de la Raspberry PI, ya que en ella se establecerá una función que permita la selección de las operaciones matemáticas por medio de la lectura de los puertos virtuales.

A. Justificación.

Implementar los conocimientos adquiridos en la asignatura Arquitectura de computadoras.

B. Objetivos específicos

- Comprender todos los conceptos más básicos de programación orientada a objetos y el uso de las librerías GPIO y time.
- Ejecutar los dos ejemplos propuestos en uno solo con el uso de funciones y la lectura de los puertos virtuales GPIO.

II. RASPBERRY PI

A. Raspberry Pi

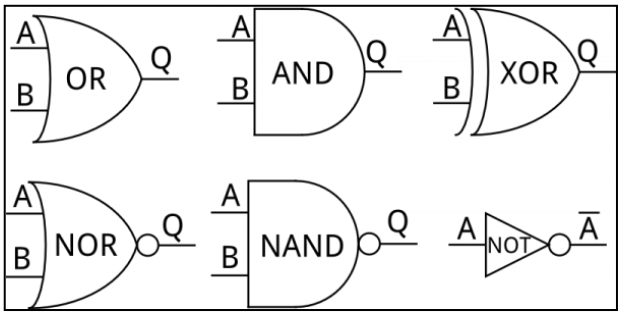
La Raspberry Pi es una computadora en una sola tarjeta (Board Computer) creada por la Raspberry Pi Foundation para promover la enseñanza de la programación en escuelas y países en desarrollo. Python es un lenguaje de programación de alto nivel, interpretado, permite programar usando los paradigmas de programación orientada a objetos o funcional y se puede extender fácilmente por medio de módulos escritos en C o C++. La Raspberry Pi se puede usar en proyectos de electrónica y para tareas básicas que haría cualquier ordenador de sobremesa como navegar por internet, hojas de cálculo, procesador de textos, reproducir vídeo en alta definición e incluso jugar a ciertos juegos. Al utilizar Python en la Raspberry Pi tenemos la ventaja de poder utiliza los pines GPIO para conectar el mundo digital con el mundo físico mediante la electrónica y programación. [3]

B. Puertos GPIO.

GPIO es un sistema de entrada/salida de propósito general que cuenta con interruptores que permiten activar o desactivar los 40 pines. Estos pines están incluidos en todos los modelos de Raspberry Pi. Los pines GPIO tienen funciones específicas (aunque algunos comparten funciones) y se pueden agrupar de la siguiente manera. Los GPIO representan la interfaz entre la Raspberry Pi y el mundo exterior. Y con ellos podrás hacer multitud de proyectos, desde encender un LED hasta otros mucho más sofisticados. Pero para eso debes saber sus características y como se programan. Lo primero variará en función de la revisión de placa que tengas o del modelo. [2]

III. Lógica combinacional

Los elementos que constituyen los circuitos digitales se caracterizan por admitir sólo dos estados. Es el caso por ejemplo de un conmutador que sólo puede estar ENCENDIDO o APAGADO, o una válvula hidráulica que sólo pueda estar ABIERTA o CERRADA. Para representar estos dos estados se usan los símbolos '0' y '1'. Se denomina lógica combinacional a todo sistema digital en el que sus salidas son funciones exclusivas del valor de sus entradas en un momento dado, sin que intervengan en ningún caso estados anteriores de las entradas o de las salidas. Las funciones (OR, AND, NAND, XOR) son booleanas donde cada función se puede representar en una tabla de la verdad. Por tanto, carecen de memoria y de retroalimentación



IV. Ejemplo

Se importarán las librerías, se definirán los pins de salida y definiremos el significado de las variables. Esto se hará para las dos clases, la alarma de incendios y el sistema de riego automático.

```
#Importar modulos
import RPi.GPIO as GPIO
import time

t=1

GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(19, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)

#Definiciones
#Puerto 3 = gas = x
#Puerto 5 = humo = y
#Puerto 7 = tem45 = z
#Puerto 11 = tem60 = w
```

```
#Definiciones
#Puerto 3 = Tanque vacio = v
#Puerto 5 = Es de dia = d
#Puerto 7 = Restricciones(es verano) = r
#Puerto 11 = Tierra seca = s
```

A) Alarma de incendios

Se creará la clase y definiremos las funciones que vamos a utilizar en el programa principal.

```
#clase
class Alarma:
  a=0
  x=0
  y=0
  z=0
  w=0

  def menu(self):
    print("\nINFORMACIÓN")
    print("3) Sensor de gas")
    print("5) Sensor de humo")
    print("7) Temperatura 45°")
    print("11) Temperatura 60°")

  def gas(self):
    x=0
    if GPIO.input(3) == GPIO.HIGH:
      x=1
    return x
  def humo(self):
    y=0
    if GPIO.input(5) == GPIO.HIGH:
      y=1
    return y
  def tem45(self):
    z=0
    if GPIO.input(7) == GPIO.HIGH:
      z=1
    return z
  def tem60(self):
    w=0
    if GPIO.input(11) == GPIO.HIGH:
      w=1
    return w
  def incendio(self,x,y,z,w):
    a = w + x*z + x*y + z*y
    return a
```

En la línea número 55 se creará la función alarma que servirá para simular el incendio, en cuando debe prender la alarma. Se utilizará operaciones ya antes analizadas para dar dicho resulto de forma efectiva.

```
59    def imprimir(self,a):
60      print("\n\n********SE QUEMA*********")
61      print("\n\nApagar Alarma pin 8")
62      print("Primero desactive los anteriores pins")
63
64    def foco(self):
65      while (GPIO.input(8) != GPIO.HIGH):
66        GPIO.output(18, True)
67        GPIO.output(19, True)
68        GPIO.output(16, True)
69        GPIO.output(13, True)
70        GPIO.output(15, True)
71        time.sleep(0.5)
72        GPIO.output(18, False)
73        GPIO.output(19, False)
74        GPIO.output(16, False)
75        GPIO.output(13, False)
76        GPIO.output(15, False)
77        time.sleep(0.5)
```

Se creará la función imprimir la cual dará la opción de apagar la alarma siempre y cuando se haya corregido los accidentes.

B) Circuito de riego automático

En el sistema de riego automático de igual manera se crearán las funciones necesarias.

```
85  #clase
86  class Bomba:
87    b=0
88    v=0
89    d=0
90    r=0
91    s=0
92
93    def menu(self):
94      print("\nINFORMACIÓN")
95      print("3) Tanque vacio")
96      print("5) Es de dia")
97      print("7) Es verano")
98      print("11) Tierra seca")
99
100   def vacio(self):
101     v=0
102     if GPIO.input(3) == GPIO.HIGH:
103       v=1
104     return v
105   def dia(self):
106     d=0
107     if GPIO.input(5) == GPIO.HIGH:
108       d=1
109     return d
110   def verano(self):
111     r=0
112     if GPIO.input(7) == GPIO.HIGH:
113       r=1
114     return r
115   def seco(self):
116     s=0
117     if GPIO.input(11) == GPIO.HIGH:
118       s=1
119     return s
```

En esta clase cambiaremos unas variables, las negaremos ya que en la resolución por mapas de karnaugh las variables son negadas.

```
120   def agua(self,v,d,r,s):
121     if v==1:
122       v=0
123     elif v==0:
124       v=1
125     if d==1:
126       d=0
127     elif d==0:
128       d=1
129     if r==1:
130       r=0
131     elif r==0:
132       r=1
133     b = v*s*(d+r)
134     return b
```

Después al igual que el sistema de alarma se creará el que hacer para apagar el sistema de riego y los focos que demuestres que se está realizando el riego.

```
136   def imprimir(self,a):
137     print("\n\n********SE ESTA REGANDO EL AGUA********")
138     print("\n\nDetener Riego pin 8")
139     print("Primero desactive los anteriores pins")
140
141   def foco(self):
142     while (GPIO.input(8) != GPIO.HIGH):
143       GPIO.output(18, True)
144       GPIO.output(19, True)
145       GPIO.output(16, True)
146       GPIO.output(13, True)
147       GPIO.output(15, True)
148       time.sleep(0.5)
149       GPIO.output(18, False)
150       GPIO.output(19, False)
151       GPIO.output(16, False)
152       GPIO.output(13, False)
153       GPIO.output(15, False)
154       time.sleep(0.5)
```

C) Función principal

En la función principal dejaremos a escoger cual sistema desea ejecutar, el de riego automático o la alarma de incendios, dependiendo cual escoja realizara lo correspondiente.

```
#Funcion principal
alam=Alarma()
riego=Bomba()
print("***BIENVENIDO***")
print("**Selecciones el pin 10 para alaram
print("**Selecciones el pin 12 para circui
while (t!=0):
  q=1
  if((GPIO.input(10) == GPIO.HIGH)):
    alam.menu()
    while (q!=0):
      gas=alam.gas()
      humo=alam.humo()
      tem45=alam.tem45()
      tem60=alam.tem60()
      si=alam.incendio(gas,humo,tem45,tem6
      if si >= 1:
        alam.imprimir(si)
        alam.foco()
        q=0
  if((GPIO.input(12) == GPIO.HIGH)):
    riego.menu()
    while (q!=0):
      vacio=riego.vacio()
      dia=riego.dia()
      verano=riego.verano()
      seco=riego.seco()
      si=riego.agua(vacio,dia,verano,seco)
      if si >= 1:
        riego.imprimir(si)
        riego.foco()
        q=0
```

V. SIMULACION EN PROTEUS

Primer ejercicio.

A continuación, utilizamos el conocimiento adquirido en la materia de sistemas digitales para implementar el circuito por medio de otra alternativa que es le herramienta y un simulador de protoboard. Para esto, una vez realizado el análisis y creado una tabla de verdad, simplificamos la sumatoria de min términos por medio del método grafico de Mapa de Karnaugh.

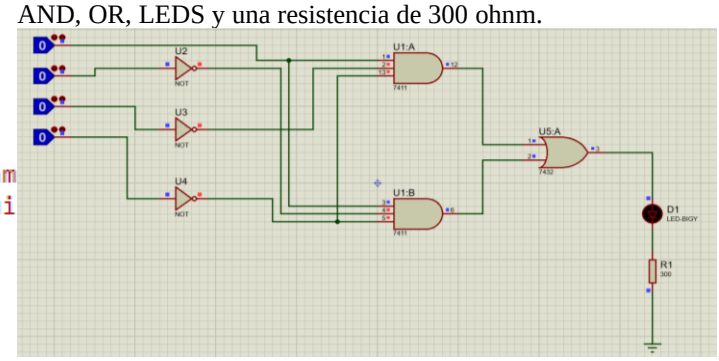
	S	R	D	V	B
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

MAPA DE KARNAUGH

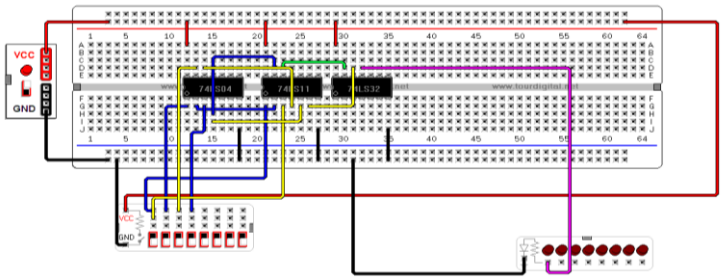
DV \ SR	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

$F(S, R, D, V) = S\bar{D}\bar{V} + S\bar{R}\bar{V}$

Una vez hecho esto podemos implementar el circuito en proteus considerando la forma simplificada que obtuvimos. Para esto utilizaremos las compuertas logicas NOT(inversor), AND, OR, LEDS y una resistencia de 300 ohnm.



Para tener una referencia mas aproximada de la forma en que se implementaria este circuito de forma tangible, utilizaremos un simulador de protoboard. Para construir nuestro corcuito en este simulador debemos considerar las fichas tecnicas de todos los circuitos integrados(7404,7411,7432).



Segundo ejercicio.

Para el segundo ejercicio repetimos el mismo proceso con la diferencia que en el análisis utilizaremos la forma POS para implementar, esto debido a que nos conviene tomar en cuanta los ceros en la salida por su cantidad.

	G	T45	T60	H	A
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	X
3	0	0	1	1	X
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	X
11	1	0	1	1	X
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

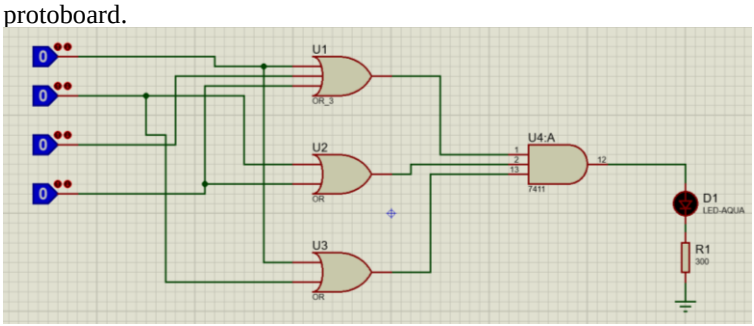
En las combinaciones T45=0, T60=1 no son posibles ya que la temperatura no puede ser menor a 45 y mayor a 60 al mismo tiempo.

MAPA DE KARNAUGH

T60 H \ G T45	00	01	11	10
00	0	4	12 X	8 X
01	1	5	13	9
11	3	7	15	11
10	2	6	14 X	10 X

F (G, T45, T60, H) = (G+T60+H). (T45+H). (T45+G)

En este ejercicio solo realizaremos la simulacion en proteus por la falta del circuito integrado para hacer uso de la compuerta or de 3 entradas en el constructor visual de protoboard.



VI. Conclusiones

- El lenguaje de programación Python es un lenguaje muy completo y gracias a la pagina creat with code que nos permite programar en dicho lenguaje y simular puertos GPIO de la Rasberry resulta muy útil.
- Existe un sinnúmero de herramientas y aplicaciones que permiten previsualizar y simular nuestros proyectos durante todo el proceso de desarrollo, estas en su mayoría requieren conocimientos básicos en el área de la programación y la electrónica ofreciendo un entorno interactivo mediante el cual se puede dar rienda suelta a la imaginación y creatividad.

VII. Recomendaciones

- Conocer o investigar acerca de programas ya realizado para obtener una mejor guía sobre la utilización tanto de la pagina como del lenguaje de programación.
- Al momento de realizar el salto para implementar de forma física nuestros proyectos debemos respetar los estándares eléctricos y revisar las fichas técnicas de los dispositivos que se van a utilizar, pues los simuladores y las herramientas en línea no siempre consideran las variables físicas que intervienen en este proceso. De esta manera podemos evitar causar daño a nuestros dispositivos o a nosotros mismos a causa del uso incorrecto de estos.
- Mantener en práctica el uso del paradigma de la programación orientada a objetos, ya que hoy en día esta es la forma más habitual y eficiente mediante la cual podemos controlar hardware por medio de un lenguaje de alto nivel.

VIII. Bibliografía

[1] N. F. K. K. I. M. M. I. ,.-C. K. Rahardhita Widyatra Sudibyo, «A TCP Fairness Control Method for Two-Host Concurrent Communications in,» Okayama University, 2019. [En línea]. Available: https://ieeexplore.ieee.org/document/8726907. [Último acceso: 06 09 2020].

[2] D. A. J. Ruby Roselin A, «Smart Agro System Using Wireless Sensor Networks,» Applied Electronics, Master of Engineering SSN College of Engineering , julio 2017. [En línea]. Available: https://ieeexplore.ieee.org/abstract/document/8250751. [Último acceso: 29 08 2020].

[3] I. d. M.-V. R.-G. S.-. A. Miguel F. Arriaga-Gomez, «A comparative survey on supervised classifiers for face recognition,» Universidad Politecnica de Madrid , 2014. [En línea]. Available: https://ieeexplore.ieee.org/document/6987036. [Último acceso: 29 08 2020].