ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Circuito control de riego - Circuito Alarma

TECHNOGEEKS
Come To Code To Lead...

Arquitectura de Computadoras

Alumnos:

Kevin Topón

Andrés Paspuel

Henry Simba

Docente:

Ing. Darwin Omar Alulema Flores

CONTENIDO

Título: Circuito control de riego -Circuito Alarma

1.- Planteamiento del Problema

Existen diversas técnicas en la actualidad, el sistema de riego automático y la alarma de incendio es cada vez más frecuente gracias a los múltiples beneficios que presenta como por ejemplo poder evitar un incendio con los conocimientos adquiridos de la materia de Arquitectura de Computadoras desarrollaremos en un emulador estos dos proyectos muy interesantes.

2.- Objetivos

Objetivo General

Desarrollar un circuito de riego y de una alarma de incendios aplicando los conocimientos ya adquiridos juntos a la herramienta de programación Python, de tal manera lo ejecutaremos en un simulador de la Raspberry la realización de estos proyectos nos permitirá enriquecer nuestros conocimientos con respecto al desarrollo de programas de este tipo.

Objetivos Específicos

-Analizar cómo funciona un sistema de riego y de una alarma de incendios para que de esta forma se pueda desarrollar correctamente los proyectos con los requerimientos solicitados .

-Conocer las características y herramientas que brinda el lenguaje de programación Python para la realización de este proyecto.

-Utilizar los pines de la Raspberry para dirigir los datos de entrada como por ejemplo el estado de la bomba etc .

3.- Estado del Arte

- En el año 2018 en Manabí, Jacqueline Castro hizo una investigación del Diseño de un entrenador con sensores mediante la plataforma raspberry, cuyos resultados fueron que se diagnosticó la falta de la plataforma Raspberry pi en conjunto con la utilización de sensores en la asignatura de Robótica

- En el año 2016 en la ciudad de Barcelona, Alberto Tobajas hizo un estudios sobre el Diseño e implementación de una estación meteorológica con Raspberry Pi y sus resultados fueron de que La dificultad se ha encontrado principalmente en los sensores analógicos principalmente en el anemómetro y el pluviómetro, los cuales entre las pocas especificaciones que se encuentran en sus hojas de características.

- En el año 2019 en la ciudad de San Luis, Argentina, Carlos Lombardi y Nicolás Passerini hicieron un estudio sobre Wollok: reconciliando didáctica e industria en

un lenguaje educativo para POO y la conclusión sacada de este trabajo fue la de presentar un lenguaje de programación educativo con varias características que facilitan una estrategia en la que los conceptos principales de la POO.

4.- Marco Teórico

Python

Según (Rossum, 1995) Python es un lenguaje de programación simple pero potente e interpretado diferente a la programación en C y Shell ideal para la programación desechable y la creación rápida de prototipos. Su sintaxis se elabora a partir de construcciones tomadas de una variedad de otros idiomas, el intérprete de Python se amplía fácilmente con nuevas funciones y tipos de datos implementados en C. Python de igual manera es adecuado como un lenguaje de extensión para aplicaciones C altamente personalizables para varios sistemas operativos como por ejemplos UNIX (incluido Linux), el sistema operativo Apple Macintosh, MSDOS describiéndolo a esta sintaxis y la semántica central del lenguaje como consiso pero intenta ser exacto y completo.

Según (Aguilar, 2010) argumenta que en Phyton se puede extender fácilmente por medio de modulos escritos en C o C++, proponiendo un énfasis en que el código sea legible (import this).

De igual manera (Duque, 2009) argumenta que Python es un lenguaje de programación creado por Guido Van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje interpretado o de script, con tipiado dinámico, fuertemente tipiado, multiplataforma y orientado a objetos.

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo, los lenguajes interpretados son más flexibles y más portables. Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera

vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

Tipado dinámico

La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Orientado a objetos

La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

Python es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, hacen que desarrollar una aplicación en Python sea sencillo

Raspberry Pi

Según (Aguilar, 2014) la Raspberry Pi es una computadora en una sola tarjeta (Single-Board Computer) creada por la Raspberry Pi Foundation para promover la enseñanza de la programación en escuelas y países en desarrollo.

Módulo time

import time

time.time() regresa el tiempo transcurrido en segundos desde el primero de enero de 1970 como un número de punto flotante

time.sleep() Suspende la ejecución del Script por el tiempo especificado como parámetro (número de segundos expresado como número de punto flotante)

Biblioteca para acceso a GPIO

>>> import RPi.GPIO as GPIO #importa la librería de GPIO

#usar número de terminal no de GPIO

>>>GPIO.setmode(GPIO.BOARD)

#coigura como salida en bajo a la terminal 11
nf>>>GPIO.setup(11,GPIO.OUT,GPIO.PUD_OFF,GPIO.LOW)

>>>GPIO.output(11,GPIO.HIGH) #pone la salida en alto

Estructuras de control básicas

Las estructuras de control más comunes son if, if - else y while, que son controladas por una condición lógica, separada del bloque controlado por : El inicio y el fin del bloque controlado depende únicamente de la indentación

Tips básicos de Python

- Importante: Python es sensible a mayúsculas y minúsculas
- Numerico: int (1 45 -678), float (12.234 -43.56), complex (-1.23+34.9j 56.1-156j)
- Cadenas de caracteres: “ Hola”, ‘Mundo’
- // división de punto flotante: 3/2=1.5
- // división entera: 3//2=1
- type(variable) regresa el tipo de variable

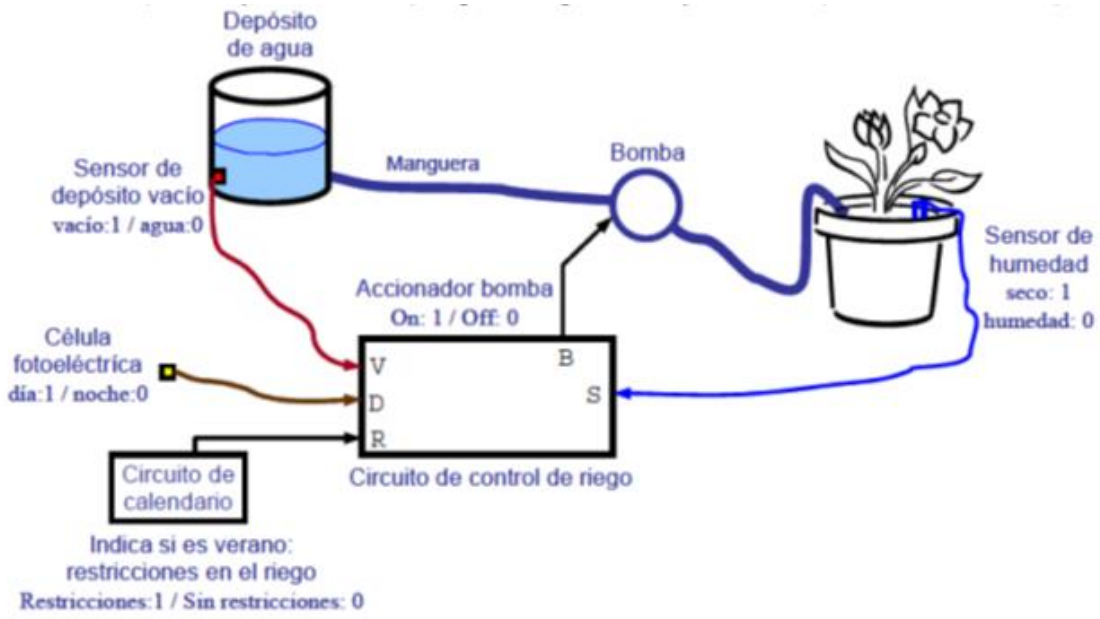
Funciones

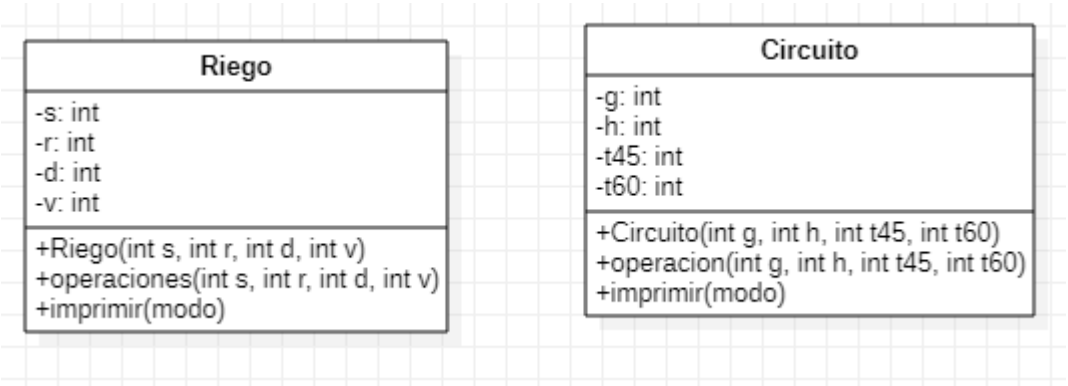
Una función de un bloque de código organizado, probado y reutilizable. Permiten que los programas sean modulares.

Python incluye muchas funciones internas, tal como printf, pero también se puede construir funciones propias personalizadas, llamadas funciones definidas por el usuario

La definición de una función comienza por la palabra clave def seguida por el nombre de la función y paréntesis Los parámetros de entrada se colocan entre los paréntesis-

5.- Diagramación





6.- Lista de Componentes

- Pines de la RASPBERRY
- Computadora
- Lenguaje de Programación PYTHON

7.- Mapa de variable

➢ import: se utiliza para importar un módulo en este caso fue time y RPi.GPIO as GPIO.

➢ self :sirve para acceder a un atributo dentro del objeto (clase) en sí.

➢ GPIO: son los puertos que emulan a la de una raspberry.

➢ setup: se define uno de los pines como entrada o salida.

➢ class: tipo de dato definido por el usuario, y la crear instancias de una clase hace relación a la creación de objetos.

➢ def: función usada para crear objetos funciones definidas por el usuario

➢ init: su función es establecer un estado inicial en el objeto nada más instanciarlo, es decir, inicializar los atributos.

➢ input:es para introducir datos de distintos tipos desde la entrada estándar.

➢ return: es el valor que nos va a devolver.

➢ if/elif/else: se utiliza para condiciones que sean más de una.

➢ print: es lo que se va a presentar en pantalla.

➢ while: repetir la condición hasta que la misma se cumpla.

8.- Descripción código fuente

Para programar hay diversas opciones pero en este caso utilizaremos https://create.withcode.uk/, entrando en detalle vamos a desarrollar nuestro programa.

Nota: los que se describe a continuación cumple con los dos enunciados planteados.

- sensor_S=3: asignamos los pines con los que vamos a trabajar.
- GPIO.setup(sensor_S, GPIO.IN): configuramos los pines de entrada.
- class riego: definimos nuestra clase y posteriormente nuestros atributos a contener.

- def __init__(self,S,R,D,V): definimos nuestro constructor.

- def operacion(self,S,R,D,V): se define cada función según nosotros le pongamos en nombre y la acción que esta va a realizar.

- elif modo==1:
  print("BOMBA ENCENDIDA: con restriccion")
  print("V=1 Hay agua")
  print("D=0 Es de NOCHE")

  Aquí se define como se va a presentar en pantalla el mensaje e la operación.

9.- Descripción de pre-requisitos

Para utilizar una las raspberry necesitamos la las librerías siguientes antes de programar:

- import time: sirve para intervalos de tiempo son números de punto flotante en unidades de segundos.

- import RPi.GPIO as GPIO: tenemos que importar la librería que nos permite manejar los GPIO.

10.- Aportaciones

En el sistema de riego se realizó de otra manera de ingresar los datos en otras palabras los datos son ingresados por consola.

11.- Conclusiones

- Con el procedimiento correcto, se pudo conectar cada parte que deseábamos y software (Lenguaje de programación Python) para realizar una simulación de un circuito de riego puede marcar la pauta a la implementación en un proyecto real.

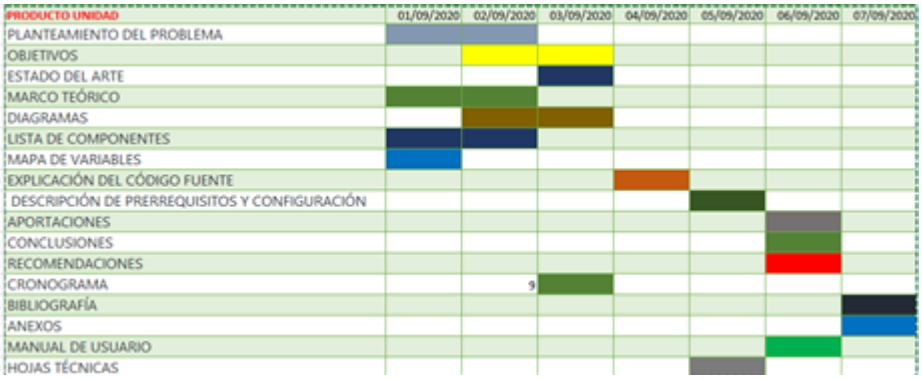
- Se ha logrado que la simulación funcione con efectividad, adquiriendo así conocimiento sobre el tema y la comprensión del funcionamiento de este sistema, agregando que las simulaciones que realiza son aleatorias y así poder observar como es que se comportaría en situaciones reales.

- Cabe recalcar que lo bueno del desarrollo de la tecnología, es que existe una diversidad de plataformas que operan con micro controladores disponibles para la computación a nivel físico (hardware). Todas estas herramientas se organizan en paquetes fáciles de usar para minimizar el trabajo del desarrollo a nivel de programación (software).

12.- Recomendaciones

- Se recomienda hacer el uso correcto de los pines GPIO de la raspberry Pi, pues cada uno está destinado para una función determinada, pues si no se realiza el uso adecuado de los mismos, podríamos tener daños como que no nos funcione el circuito.
- Antes de empezar a utilizar conceptos POO se debe informar y plantear de forma clara el objetivo del proyecto, para facilitar así el uso del mismo, sin tener que ser un experto en el tema.

13.- Cronograma



14.- Bibliografía

Aguilar, M. F. (2014). Programación en Python en la. Obtenido de http://sagitario.itmorelia.edu.mx/mfraga/materias/soemb/python.pdf

Duque, R. G. (2009). Python para todos. España. Obtenido de http://www.utic.edu.py/citil/images/Manuales/Python_para_todos.pdf

OLIPHANT, T. E. (15 de 05 de 2007). Python for Scientific Computing. CiSE Computational Physics.

Rossum, G. (1995). Manual de referencia de Python. *CWI (Centro de Matemáticas e Informática)*.

Chazallet, S. (2016). *Python 3: los fundamentos del lenguaje*. Ediciones ENI.