

Documentación oficial de CEFootV.4

Rodrigo Arce^{1,†} and David García^{2,†}

¹Ingeniería en Computadores, Tecnológico de Costa Rica
²Ingeniería en Computadores, Tecnológico de Costa Rica
[†]Todos los autores contribuyeron de manera igual al trabajo

Compilado el 7 de mayo, 2024

Abstract

Internet of Things es un concepto que se ha popularizado en los últimos años. Este proyecto tiene como objetivo ponerlo en práctica mediante una Raspberry Pi Pico W. La idea del proyecto es crear una maqueta de un estadio de fútbol. Será un partido de penales, el primer equipo que meta 3 goles o que tenga más goles que el rival, gana. La detección de los goles funciona mediante la pulsación de uno de los 6 botones que están colocados detrás de la portería. El Raspberry Pi Pico W se va a conectar a una PC y va a enviar y recibir señales mediante un servidor HTTP.

Keywords: Electrónica, Computador, Circuitos electrónicos, Servidor, HTTP, Raspberry pi pico w

Corresponding author: Estudiantes de Ingeniería en Computadores del Tecnológico de Costa Rica. *E-mail address:* r.arce.1@estudiantec.cr¹, d.garcia.1@estudiantec.cr².
URL: <https://github.com/IVoidi/CEFootV.4>

MIT License

■ INTRODUCCIÓN

Internet of Things es un concepto que se ha popularizado en los últimos años. La posibilidad de poder desarrollar sistemas que se conectan entre sí, sin la necesidad de un cable de por medio ha abierto muchas posibilidades en el mundo de la tecnología. El presente proyecto busca ejemplificar el concepto de Internet of Things mediante una Raspberry Pi Pico W. Este dispositivo tiene la capacidad de conectarse mediante Wi-Fi, lo que permite desarrollar este proyecto.

■ REGLAS DEL JUEGO

Este mismo trata de un juego que busca simular una serie de penales mediante un circuito simple. Dos equipos van a competir en una tanda de cinco penales, el primer equipo que meta tres goles o meta más goles que el enemigo, es el ganador. El estadio va a ser una maqueta, y la clave del juego está en la cancha. La cancha va a contar con seis paletas. La idea es que cada paleta esté asociada a un botón que va a ser accionado por una bola.

■ FRONT-END Y BACK-END DEL PROYECTO

Para este juego, se desarrolló una interfaz gráfica hecha con Tkinter. La idea del juego es conectar la Raspberry Pi Pico W con la computadora mediante un servidor HTTP desarrollado con socket. La interfaz cuenta con la posibilidad de elegir visualmente entre tres equipos. Cada equipo cuenta con 6 jugadores, los cuales pueden ser elegidos tanto como artilleros como porteros a voluntad. La selección funciona mediante peticiones POST, las cuales son enviadas a la Raspberry Pi Pico W y se transforman en una de las tres señales: SIGTEAM, SIGPOT y SIGGOAL. Cuando se acciona la señal SIGTEAM, va a ejecutar un código que va a escuchar al botón de cambio de equipo y va a cambiar el color del led asociado.

```
1 selected = False
2 while not selected:
3     print(selected)
4     if change_player_signal.value() and
       current_team_playing == "blue":
5         current_team_playing = "red"
6         selected = True
7         blue_team.value(0)
8         red_team.value(1)
9     elif change_player_signal.value() and
       current_team_playing == "red":
```

```
10         current_team_playing = "blue"
11         selected = True
12         blue_team.value(1)
13         red_team.value(0)
```

Code 1. Muestra del código de la señal SIGTEAM.

Cabe destacar que, al conectar la Raspberry Pi a la batería, esta va a encender por defecto la luz azul. Esta representa al equipo local, mientras que la luz roja va a ser el equipo visitante. Esto no significa que el equipo local sea siempre fijo. A nivel de código, el equipo local se elige mediante una función de números pseudoaleatorios en el código de cada una de las interfaces. La siguiente señal a analizar es la señal SIGPOT. Esta es la señal más simple de todas, pues lo que va a hacer es enviar un valor del potenciómetro que va a estar comprendido entre 0 y 6. Y por último, está la señal SIGGOAL. Esta señal va a esperar a que una de las 6 paletas sea activada, y además va a crear el algoritmo de anotación.

```
1 anotation_algorithm = random.randint(1, 3)
2
3 # Indices de las paletas en las que esta el
  portero
4 index_list = []
5 if anotation_algorithm == 1:
6     goalkeeper_index = random.choice(goal)
7     index_list = [goalkeeper_index,
8                   goalkeeper_index + 1] if goalkeeper_index +
9                   1 < len(goal) else [goalkeeper_index - 1,
10                                     goalkeeper_index]
11 elif anotation_algorithm == 2:
12     goalkeeper_index = random.choice(goal)
13     index_list = []
14     if goalkeeper_index == len(goal) - 1:
15         index_list = [goalkeeper_index-2,
16                       goalkeeper_index-1, goalkeeper_index]
17     elif goalkeeper_index == 0:
18         index_list = [goalkeeper_index,
19                       goalkeeper_index + 1, goalkeeper_index + 2]
20     else:
21         index_list = [goalkeeper_index-1,
22                       goalkeeper_index, goalkeeper_index + 1]
23 else:
24     group = random.randint(0, 1)
25     index_list = []
26     if group == 1:
27         index_list = list(filter(lambda x: (x %
28                                           2 == 0), goal))
29     else:
30         index_list = list(filter(lambda x: (x %
```

```
2 == 1), goal))
```

Code 2. Muestra del código de la señal SIGGOAL.

Hay tres maneras distintas en las que el programa escoge el portero. La primera manera que se planteó es que agarre un par de las paletas, la segunda manera es que agarre un trío de paletas juntas y la última manera es que agarre uno de los grupos de las paletas intercaladas. La selección de este algoritmo es pseudo-aleatoria y no es visible para el jugador, pues sucede del lado del servidor. Todos estos datos son enviados como respuesta a la solicitud enviada por el cliente a la Raspberry Pi.

```
1 addr = socket.getaddrinfo('0.0.0.0', 8080)
2 [0][-1]
3 s = socket.socket()
4 s.bind(addr)
5 s.listen(1)
6 s.settimeout(100)
7 print(f"[DEBUG] Escuchando al puerto 8080, en {
8     addr}")
```

Code 3. Creación del servidor.

Después de crear con éxito el socket, podemos recibir señales por parte de un cliente. Al recibir las señales y ejecutar las instrucciones designadas, se envía la señal mediante la función `send_code`.

```
1 def send_code(code, client, address, timeout=0):
2     if timeout > 5:
3         cl_file = client.makefile('r')
4         while True:
5             line = cl_file.readline()
6             if not line or line == b'\r\n':
7                 break
8         print(f'enviando {code}')
9         client.send('HTTP/1.0 200 OK\r\n\r\n')
10        print('enviando dato')
11        client.sendall(code.encode())
12        print('cerrando')
13        client.close()
```

Code 4. Código para enviar código al cliente.

■ SOBRE LAS ANIMACIONES

Este código se caracteriza por tener una gran cantidad de animaciones. Desde animaciones más simples como la animación de la moneda o la animación de la explosión, hasta la animación del disparo de la bola. La animación de tanto la explosión como de la moneda, son lineales. Simplemente recorren una lista con cada uno de los frames. Sin embargo, la animación del disparo de la bola es un poco más compleja que esto.

La animación de esta bola está dividida en dos tracts, los cuales, entre ambos, suman 1,50s. Para la primera parte, la bola se dirige a un punto específico. Este punto está definido por la diferencia entre el punto de referencia y el punto final, sumado con una desviación de nombre *bias* en el código.

```
1 new_position = (reference[0] + (difference_x +
2     bias) * ratio,
3     reference[1] + (
4     difference_y - 100) * ratio)
```

Code 5. Posición en función del tiempo de la bola en el primer tracto.

Este movimiento está acelerado por la función llamada `ratio`. Esta función devuelve un número entre 0 y 1, basado en el punto actual de la bola.

$$r(t) = -1,778 \cdot t(t - 1,5)$$

Sin embargo, cabe recalcar que el movimiento de la bola **no corresponde a movimiento de proyectil**, sino a un simple movimiento rectilíneo uniformemente acelerado. Este movimiento de *MRUA*

sucede por 0,75s.

El segundo transecto sucede después de la primera mitad, en este momento, la bola cambia su punto de referencia. Esto es necesario ya que la bola se requiere que llegue a un nuevo punto el cual será el centro de la paleta. Si se mantenía el punto de referencia anterior, la bola regresaría al punto inicial de disparo. Entonces, para este segundo tracto, se sigue el mismo concepto pero de manera *"inversa"*.

```
1 new_position = (last_pos[0] + difference_x * (1
2     - ratio),
3     last_pos[1] +
4     difference_y * (1 - ratio))
```

Code 6. Posición en función del tiempo de la bola en el segundo tracto.

■ DIFICULTADES ENCONTRADAS

Aunque la Raspberry Pi Pico W sea un dispositivo muy útil, puede presentar problemas a la hora de armar un servidor en el mismo. La Raspberry Pi presentó problemas a la hora de recibir las señales. Específicamente, la señal SIGGOAL causó problemas, ya que el output solía ser el mismo: `ConnectionResetError`. Al parecer el error era causado por el tamaño de buffer recibido por el servidor, además de que la solicitud del cliente no era leída en sí, solo se estaba leyendo el largo de la petición y esto causaba los problemas.

■ CONCLUSIONES

Se logró desarrollar un ambiente simulado mediante circuitos electrónicos y bajo el concepto de Internet of Things. La Raspberry Pi Pico W ha demostrado ser un dispositivo útil que tiene capacidad de montar un servidor simple para la facilitación de algunas tareas, como la conexión PC-Maqueta. Se ha desarrollado, con éxito, un servidor HTTP capaz de recibir peticiones POST y de responder a ellas en la mayoría de veces. A su vez, se ha logrado integrar este backend con una interfaz gráfica en Tkinter, la cual lleva el marcador del juego y tiene la capacidad de seleccionar al portero y al artillero para cada tiro.