SICC

(Sistema Instantáneo de Censo de Corredores)

SuSoft

Autores: Agustín Ezequiel Maio, Tomás Canzoniero.

Índice

- Nombre del proyecto: SICC (Sistema instantáneo de censo de corredores)
- Integrantes, detallando nombre y apellido, DNI, curso: Canzoniero Tomas 43309977 5B, Maio Agustín Ezequiel 42819335 5B
- Docentes responsables y/o tutores: Gabriel Alejandro Jiménez
- Fecha de inicio: 20/04/2017
- Duración, en semanas: 20
- Esfuerzo en horas, considere el tiempo total: 770hs (Horas canzoniero + horas Maio)
- Personas afectadas, promedie la cantidad de personas afectadas a lo largo del proyecto indicando un tiempo promedio de dedicación de horas: Participaron 5 personas en un promedio de 60 horas a lo largo de 20 semanas:

Objetivos:

Hoy en día las competiciones de ciclismo no poseen un sistema seguro y rápido para el control de las posiciones de los competidores. Esto nos llevó a pensar en mejorar el sistema actual. Nuestro objetivo es automatizar el proceso de inscripción, aumentando la fiabilidad de los datos e incrementado el manejo de los mismos. También nos preocupamos por el censo de las posiciones a la hora de que el corredor cruza la línea de meta.

La fase de inscripción actual puede ser un tanto tediosa para el corredor y muchas veces se generan errores humanos ya que los organizadores necesitan realizar manualmente muchas tareas que nosotros pretendemos automatizar. Nuestra web personalizable a pedido del organizador permitiría al corredor inscribirse y pagar la carrera para que al momento de correr no se pierda tiempo con papeleo innecesario.

SICC pretende otorgar a cada participante un dispositivo único de identificación que irá montado en una posición estratégica de la bicicleta para el desarrollo óptimo del sistema. Este dispositivo se encargara, de que al cruzar la línea de meta, los datos sean enviados para su posterior procesamiento.

Nuestro sistema pretende mostrar a través de una aplicación web sencilla e intuitiva, los datos de los corredores de una manera instantánea y así facilitar a los corredores y sus familias la consulta de los datos con una gran rapidez.

Otra problemática que surge en estas competencias es la falta de puntos de control (ya sea por falta de infraestructura o de personal humano). Podrían ocurrir dos inconvenientes. El primero recae en la posibilidad de que el participante sufra un accidente. De ser así, el mismo no cuenta con la debida ayuda para abordar el caso. El segundo inconveniente nace cuando el corredor acorta camino o realiza algún tipo de trampa. Si bien se apela a la honestidad de este, es una situación posible que debe ser considerada y solucionada.

Al ser un sistema automatizado, ambos problemas son fáciles de tratar. En el primer caso, se acopla un botón de pánico en el dispositivo para ser activado ante cualquier emergencia. En el segundo caso, se agregan puntos de control cada determinada distancia, los cuales son obligatorios cruzar para validar la llegada a la línea de meta.

Especificaciones técnicas:

- Base de datos: MySQL.
- Plataforma web:
    - Tecnologías utilizadas:
        - Python 2.7.
        - Flask.
        - HTML5.
        - CSS3.
        - Bootstrap.
        - JavaScript.
        - jQuery.
        - API Mercadopago.
        - Adobe Photoshop.
        - Hosting web HELIOHOST.
        - Servidor apache.
    - Interfaz creadora de carrera:
        - Intermediario entre los datos obtenidos por cada corredor y el servidor web donde son publicados los mismos.
- Sistema operativo basado en Debian Linux.
- Arduino UNOr3.
- Arduino Nano.
- PC o raspberry pi.
- NRF24L01.
- Modulo IR vs1838b.
- Lenguaje de programación C.

Temática:

Deportiva. Organización de competencia.

Alcance, tanto social como geográfico:

En cuanto a lo geográfico, se extiende a nivel nacional. En cuanto a lo social, encontramos baja, media y alta competencia.

Segmento destino (clientes):

Tomando de ejemplo competencias de alto nivel (Rio Pinto, Trasmontaña, Rally de Tandil, etc.) nos encontramos con determinadas problemáticas a la hora de procesar y analizar los grandes volúmenes de datos que se manejan en estos eventos. El corredor atraviesa un largo periodo de ardua espera para conocer los resultados obtenidos en la carrera. Así mismo, el sistema actualmente empleado no es totalmente fiable ya que se podrían producir errores y confusión. SICC está orientado a pequeñas, medianas y grandes maratones garantizando en todo momento la fiabilidad de los datos, proveyendo un sistema veloz capaz de manejar correctamente la información.

Ámbito de incumbencia (deportivo, social, comunitario):

Deportivo.

Descripción general:

El proyecto SICC (Sistema Instantáneo de Censo de Corredores), tiene como propósito censar, analizar e informar en tiempo real grandes volúmenes de datos y estadísticas de corredores de ciclismo. El sistema multiplataforma contempla desde la fase de inscripción hasta el cierre de cómputos y su posterior informe online. Además permite ver un histórico dinámico para su análisis estadístico.

Funcionalidades:

- Censo de corredores.
- Actualización online en tiempo real de los datos censados.
- Intuitivo sistema de inscripción.
- Sistema de procesamientos de pagos.
- Botón S.O.S. (Por accidentes).
- Puntos de control.
- Base de datos para el correcto manejo de la información.
- Sistema de mensajería para comunicación 24hs con el organizador.
- Aprovisionamiento del centro de cómputos y el hardware de identificación única por corredor.
- Mantenimiento del hardware provisto.

Beneficios:

- Fiabilidad de los datos.
- Velocidad de procesamiento.
- No requiere trabajo humano para la inscripción de los corredores.
- Facilidad de uso.
- Interfaz de usuario intuitiva.
- Manejo de grandes volúmenes de datos.
- Económico.
- Personalizable a gusto del organizador del evento.
- No da lugar y soluciona acciones deshonestas por parte de los participantes.
- Rápida respuesta a imprevistos tales como accidentes.

Descripción técnica:

Software

La creación del sistema comenzó a partir de la necesidad presentada por un individuo activo de las carreras de ciclismo. Se procedió a hacer un relevamiento exhaustivo sobre la situación actual de dichas competiciones que involucran a varias organizaciones. Luego se propuso una solución que fue aceptada por especialistas del ámbito.

Para llevar a cabo dicha propuesta se comenzó con el modelaje de la base de datos a partir de los datos y relaciones relevadas en las entrevistas con las distintas organizaciones. Lo que se realizó fue un Diagrama Entidad Relación, que es una herramienta de modelado de datos que permite representar entidades de un sistema de información, así como sus interrelaciones y atributos. Luego se procedió a armar un Modelo Relacional que dio como resultado un conjunto de siete relaciones que representan las tablas que almacenaran los datos del sistema multiplataforma de tiempo real de acceso mundial.

Con el diseño completo de la base de datos se procedió a desarrollar la misma en lenguaje MySQL, utilizando un gestor de base de datos apropiado. Dicho gestor es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation. Para perfeccionar la creación de la base de datos se desarrolló un script en Python que permite la automatización de tareas relacionadas con el desarrollo de la base de datos.

El script se divide en distintas etapas, comenzando por la conexión a la base de datos:

```
import MySQLdb #Importamos el módulo de MySQLdb

db = MySQLdb.connect("tommy.heliohost.org", "********",
"******************", "susoft1_SistemaCarreras") #Nos conectamos indicando
HOST, USER, PASSWORD, NOMBRE DE LA BASE DE DATOS.
```

Luego se preparan todas las consultas SQL.

```
consulta1="CREATE TABLE Categoria(IdCategoria int NOT NULL PRIMARY KEY
AUTO_INCREMENT, Nombre varchar(20) NOT NULL, EdadMinima int, EdadMaxima
int);"

consulta2="CREATE TABLE Persona(NroDoc bigint NOT NULL PRIMARY KEY,
Nombre varchar(50), Apellido varchar(50), Direccion varchar(100), Email
varchar(50), FecNac date, estadoPago boolean, idPago varchar(10));"

consulta3="CREATE TABLE Corredor(IdCategoria int NOT NULL, NroDoc bigint
NOT NULL,Nro int NOT NULL,PRIMARY KEY(IdCategoria,NroDoc,Nro));"

consulta4="CREATE TABLE Provincia(IdProvincia int NOT NULL PRIMARY KEY
AUTO_INCREMENT, Nombre varchar(20) NOT NULL);"
```

```
consulta5="CREATE TABLE Lugar(IdLugar int NOT NULL PRIMARY KEY AUTO_INCREMENT, Nombre varchar(20) NOT NULL,IdProvincia int,FOREIGN KEY(IdProvincia) REFERENCES Provincia(IdProvincia) );"

consulta6="CREATE TABLE Carrera(IdCarrera int NOT NULL PRIMARY KEY AUTO_INCREMENT,Descripcion varchar(50),IdLugar int,FOREIGN KEY(IdLugar) REFERENCES Lugar(IdLugar));"

consulta7="CREATE TABLE Evento(IdEvento int NOT NULL AUTO_INCREMENT, IdCarrera int NOT NULL , IdCategoria int NOT NULL, NroDoc bigint NOT NULL,Nro int NOT NULL,Posicion int, Tiempo datetime,PRIMARY KEY(IdEvento, IdCarrera,IdCategoria,NroDoc,Nro), FOREIGN KEY(IdCarrera) REFERENCES Carrera(IdCarrera), FOREIGN KEY(IdCategoria,NroDoc,Nro) REFERENCES Corredor(IdCategoria,NroDoc,Nro));"
```

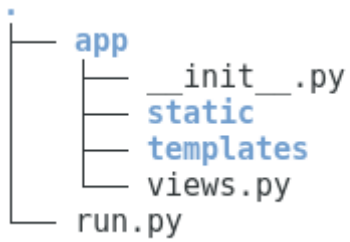
Por último, se ejecutan dichas consultas y se cierra la conexión.

```
cursor.execute(consulta1)
cursor.execute(consulta2)
cursor.execute(consulta3)
cursor.execute(consulta4)
cursor.execute(consulta5)
cursor.execute(consulta6)
cursor.execute(consulta7)
db.close() #Cerrar conexión
```

Se hizo un prototipo local para permitir verificar el correcto desarrollo de la base de datos. Se presentó ante un conjunto de usuarios quienes nos hicieron una devolución favorable que nos permitió continuar con el desarrollo de nuestra aplicación web.

La elección tomada para determinar el lenguaje en que se desarrollaría el sitio, fue Python junto al microframework Flask. Este permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene una licencia BSD.

Se requiere una estructura básica para utilizar esta herramienta. A continuación se detallará la misma.



- Run.py: Este archivo contiene la información necesaria para iniciar la aplicación.
- __init__.py: Archivo que el framework busca inicialmente para cargar la configuración.

- Views.py: Es la parte principal de la aplicación web ya que aquí se programa todo el backend de la página.
- Static: En esta carpeta se ubican los recursos necesarios para el frontend.
- Templates: Aquí se almacenan los archivos .HTML de la página web.

En la siguiente parte del informe se detallará el contenido del archivo views.py, ya que este contiene la mayoría del código del sistema web.

Inicialmente en este archivo se encuentran los módulos utilizados, tales como plugins de Flask para formularios, templates o el servicio de mail que ofrece el framework. Además se incluye el módulo de mercado pago necesario para hacer la comunicación con esta plataforma.

```
# -*- coding: utf-8 -*-
from flask import request, redirect, render_template, url_for, flash,
Flask
import forms
import os
import MySQLdb #Importamos el modulo de mysqldb
import solocaracter
import convertircategoria
import validarDNIrepetido
import mail
import mercadopago
import CatYNroCorredor
import reg
import time
```

Luego se realiza la configuración necesaria del mail y mercado pago.

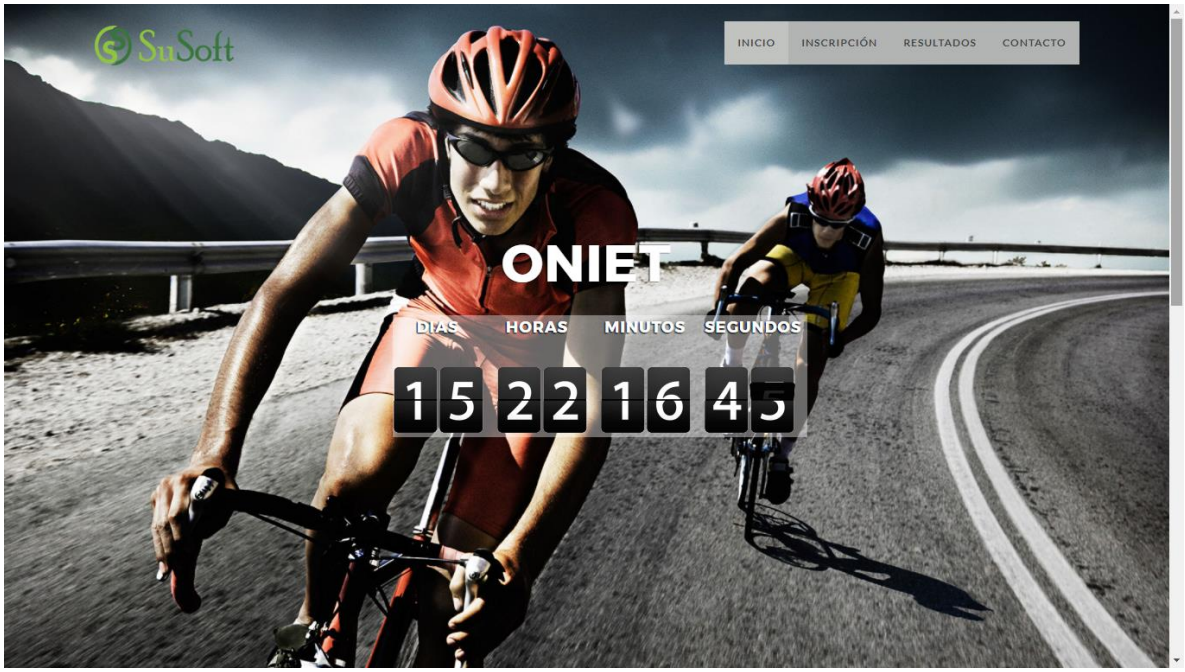
```
app.config['MAIL_SERVER']='mail.susoft.com.ar'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'no-responder@susoft.com.ar'
app.config['MAIL_PASSWORD'] = '*******'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True

sendmail = Mail(app)

mp = mercadopago.MP("5621951586369656","************************")
app.secret_key = os.urandom(24)
```

El sistema web cuenta con cuatro rutas, Inicio, Inicio Móvil, Resultados e Inscripción .Cada una de ellas fue definida de la siguiente manera.

```
@app.route('/' , methods=['GET' , 'POST'])
@app.route('/Inicio', methods=['GET' , 'POST'])
@app.route('/Inscripcion', methods=['GET' , 'POST'])
```

```
@app.route('/Resultados')
@app.route('/InicioMovil', methods=['GET' , 'POST'])
```

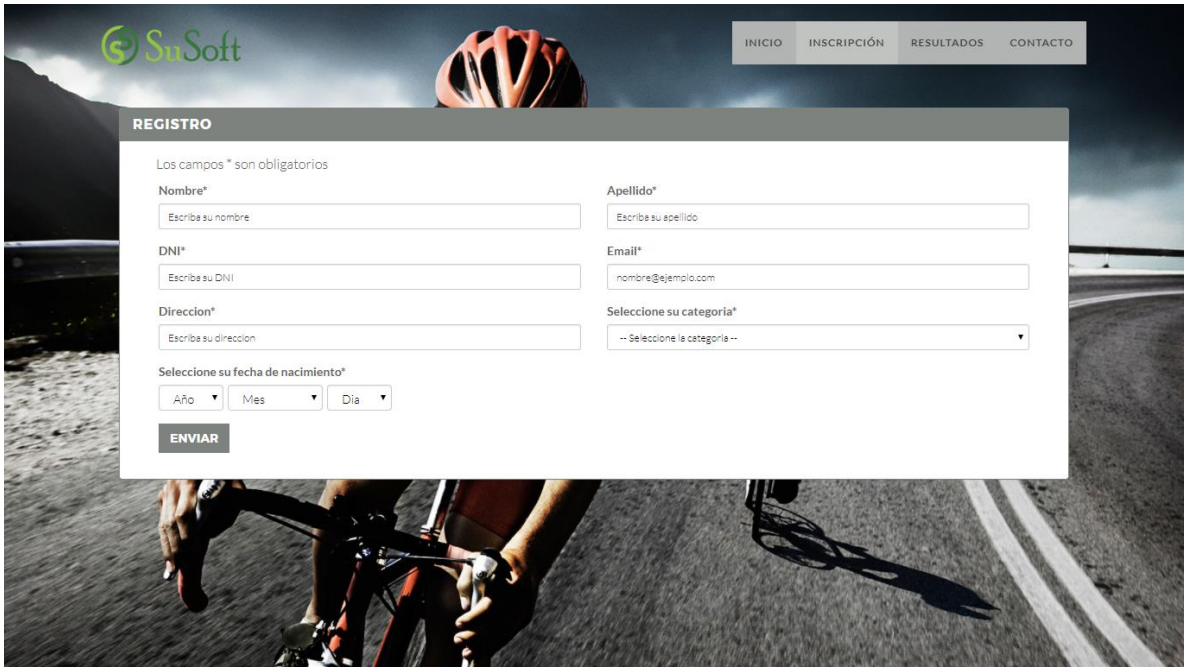
- **Inicio**: Es la página principal. Cuenta con el título de la futura carrera y además un contador con los días que faltan para el respectivo evento. En la parte inferior de esta página se encuentra la sección de contacto. Esta es utilizada para que los usuarios puedan comunicarse con nosotros en caso de algún problema.



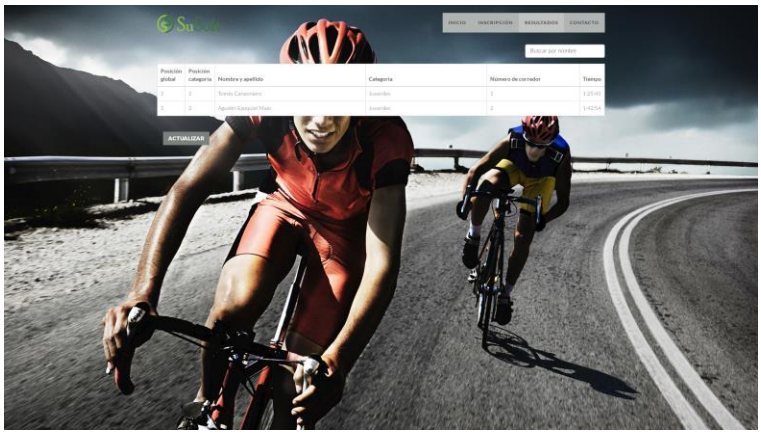
- **InicioMóvil**: Es una copia a “Inicio” pero rediseñada y optimizada para un mejor desempeño en dispositivos móviles tales como tablets o teléfonos celulares.

- **Inscripción**: Aquí los corredores se registraran en la base de datos. Se procuró que el formulario de registro sea lo más intuitivo posible, por lo tanto este requiere solo la información esencial del corredor evitando inscripciones tediosas. Además se realizan verificaciones de datos para no provocar errores en nuestra base de datos y complicar el desarrollo normal del sistema.

  Una vez registrados, los usuarios son redirigidos al sitio de Mercado Pago donde realizan la compra con todas las formas de pago que esta plataforma ofrece. Por último, es enviado un correo electrónico al usuario con un informe detallado de sus datos y el estado del pago.



- **Resultados:** Esta sección actualiza en tiempo real los resultados de la carrera. A medida que los jugadores cruzan la línea de meta, nos conectamos con la base de datos y procedemos a mostrar la posición del jugador, su nombre, la categoría y el tiempo realizado.



A continuación procedemos a detallar cada módulo empleado:

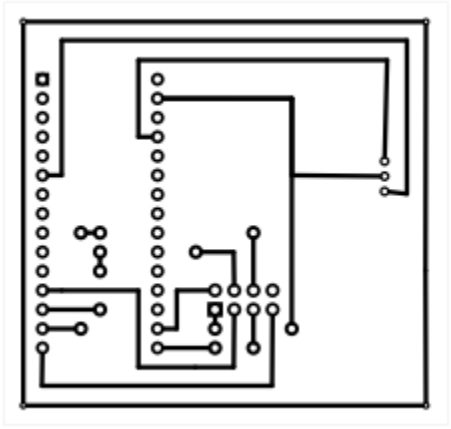
- `from flask import request`: Este módulo es el encargado de gestionar y procesar correctamente cada petición que el cliente hace al servidor.
- `from flask import redirect`: Este módulo es el encargado de redirigir a una url externa o no al servidor.
- `from flask import render_template`: Este módulo es el encargado de renderizar las plantillas HTML.
- `from flask import url_for`: Este módulo es el encargado de crear URLs dinámicas.
- `from flask import flash`: Este módulo es el encargado de enviar al cliente un mensaje.
- `from flask import Flask`: Este módulo es necesario para correr una aplicación Flask en el servidor.
- `import forms`: Este módulo gestiona y crea los formularios utilizados.
- `import os`: Este módulo es el modulo del sistema, para ejecutar comandos en el mismo.
- `import MySQLdb`: Este módulo permite conectarnos a la base de datos utilizada.
- `import solocaracter`: Este módulo filtra los campos de entrada en los formularios.
- `import convertircategoria`: Este módulo convierte el ID de la categoría a su nombre correspondiente.
- `import validarDNIrepetido`: Este módulo valida que el DNI que se está ingresando, no exista.
- `import mail`: Este módulo nos permite utilizar un sistema de mensajería por email.
- `import mercadopago`: Este módulo nos permite utilizar la API de MercadoPago
- `import CatYNroCorredor`: Este módulo convierte el ID de categoría ingresado a su nombre correspondiente, y le asigna a cada corredor un numero único.
- `import reg`: Este módulo nos permite tener un registro de las acciones realizadas por el usuario.
- `import time`: Este módulo nos permite trabajar correctamente con la hora y fecha del sistema.

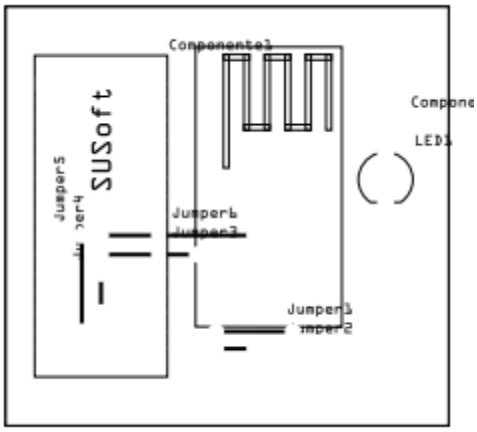
Hardware

El hardware consiste en un sensor conectado a un microcontrolador de la familia ATmega, capaz de procesar los datos recibidos para luego ser transmitidos mediante radiofrecuencia al centro de cómputos, donde se evaluara la información para su exhibición de manera ordenada.

Para lograr lo propuesto, se evaluaron diferentes métodos de censado para llevar el control del momento en que el participante de la carrera cruza la línea de meta, finalizado así el recorrido. Se optó por utilizar luz infrarroja haciendo uso del módulo vs1838b. Para transmitir los datos generados por dicho modulo se empleó el módulo nrf24l01, un chip transceptor de 2.4ghz. Por último, como microcontrolador se utilizó el ATmega328p, un circuito integrado de alto rendimiento basado en la arquitectura RISC.

En una primera instancia, se realizó el diseño del circuito y se comprobó el correcto funcionamiento del mismo. Ante la necesidad de optimizar el tamaño, se rediseñó.





El microcontrolador fue programado mediante el IDE de Arduino, desarrollando dos Sketchs que luego se le cargarían al emisor (que va montado en la bicicleta del corredor) y al receptor (conectado al centro de cómputos)

El emisor incluye las siguientes librerías:

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
```

- SPI.h: SPI Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Esta librería es la adaptación para Arduino.
- nRF24L01.h: Librería del módulo de radiofrecuencia NRF24L01.
- RF24.h: Driver para el transceptor NRF24L01

El sketch incluye dos funciones:

En la primera, setup(), se inicializó la comunicación serie y por radiofrecuencia, y se abrió un canal de comunicación.

```
Serial.begin(9600);

radio.begin();

radio.openWritingPipe(0xF0F0F0F0E1LL);
```

En la segunda, loop(), se declaró el numero único de corredor.

```
char mensaje[6] = "ID4a-1"; //Número de corredor a manera de ejemplo.
```

Y al cruzar la línea de meta, recibiendo una señal infrarroja, el mismo se emite:

```
if (digitalRead(3) > 0)
    radio.write(mensaje, largoMensaje);
```

El emisor incluye las siguientes librerías:

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
```

- SPI.h: SPI Es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Esta librería es la adaptación para Arduino.
- nRF24L01.h: Librería del módulo de radiofrecuencia NRF24L01.
- RF24.h: Driver para el transceptor NRF24L01

El sketch incluye dos funciones:

En la primera, setup(), se inicializó la comunicación serie y por radiofrecuencia, y se abrió un canal de comunicación.

```
Serial.begin(9600);

radio.begin();

radio.openReadingPipe(1, 0xF0F0F0F0E1LL);
```

En la segunda, loop(), se declaró una variable para almacenar el numero único de corredor, y se guardó en ella lo recibido.

```
char mensaje[6]; //Número de corredor a manera de ejemplo.
radio.read(mensaje, largoMensaje);
```

Por último, se envió el valor de esta variable (correspondiente al corredor que cruzó la línea de meta) por el puerto serie.

```
Serial.println(mensaje);
```

Conclusión:

Obtuvimos un innovador sistema de control y censo para carreras que permite alivianar el trabajo humano y brinda una mayor fiabilidad de los datos. A su vez el sistema otorga a los participantes una fácil y completa inscripción online que permite ahorrar tiempo al momento de la carrera.
Se busca la implementación de un sistema de posicionamiento GPS para saber en dónde se encuentra el participante en cada momento de la carrera.

Agradecimientos:

Por orden alfabético:

- Ariel Gironelli
- Gabriel Jiménez
- Luis López
- Matías Flores
- Nahuel Robles
- Pablo Ratibel
- Walter Carnero