SmartParking

Manejo de recursos escasos

Marcelo Bustos Ramírez.

Sebastián Robles González.

Cristian Vidal Sepúlveda.

Temuco, abril 2018

Índice de contenidos

Introducción

En el mundo existen más de 1,1 billones de vehículos, en Chile más de 5 millones, en la Araucanía más de 220 mil (datos entregados por el instituto nacional de estadística en el año 2016), con el rápido avance de las ciudades y del crecimiento demográfico los espacios para construir estacionamientos se han reducido radicalmente y la demanda de estos crece exponencialmente cada año. Por esta razón se vuelve cada vez más necesario un sistema que sea capaz de optimizar y facilitar su manejo.

Con esto en mente nuestro equipo de trabajo busca desarrollar un sistema que permita detectar y manejar el estado de los espacios de un estacionamiento y con ello facilitar su gestión, navegabilidad además de reducir el tiempo invertido en estacionar. Todo esto con el fin de optimizar el uso de este recurso cada vez más escaso en la sociedad actual.

Para este prototipo nos basamos principalmente en herramientas de java y arduinos, además de herramientas de diseño, para generar las primeras propuestas de interfaces gráficas, y herramientas que permitan generar u organizar calendarios de trabajo y asignar responsabilidades. Adjunto a este informe se entrega un prototipo básico del proyecto y nuestras impresiones de este.

Resumen del caso

Nombre del proyecto: *"Smart Parking (SP)".*

Nombre del producto: *"Buscando Espacio".*

Idea fundamental del proyecto:

Producir un software que permita a los usuarios conocer la disponibilidad de espacio en algún estacionamiento del sector.

¿Qué soluciona?

Optimiza el uso de un recurso escaso, el espacio disponible para estacionar el vehículo particular.

Ahorra el tiempo que invierten los usuarios en buscar un espacio disponible.

Disminuir la congestión vehicular que se genera en los estacionamientos cuando no hay cupos.

Público objetivo:

Todos los estacionamientos que implementen el software.

Usuarios que posean vehículos particulares.

Principales secciones del Software

El software, en su última versión, ha de constituirse de 3 secciones principales. Una que recolecte datos, una que interprete esos datos y una que presente la información relevante de los datos recolectados.

Módulos y métodos

Cada una de las secciones anteriores necesitará de uno o varios módulos para operar. La primera sección requiere de dos métodos, uno que “lea” el voltaje medido por Arduino y otro que pueda transferir dichas lecturas hacia el programa principal, en Java.
La segunda sección requiere de dos métodos que interprete el significado de las lecturas recibidas y otro que le entregue dichas interpretaciones a la sección final.
La sección final requiere de dos métodos, uno que despliegue la información interpretada y otro que sirva como interfaz con el usuario.

Cabe destacar que estos son los métodos planteados para un prototipo a pequeña escala, y que la versión final del Software puede requerir más métodos.

Datos gestionados por el Software

Se planea que el software muestre el estado actual de los estacionamientos. Para ello, debe tener acceso a las lecturas de los voltajes hechos por Arduino, los cuales varían dependiendo de cuánta luz es recibida por los sensores.

Diseño de interfaces gráficas

Ventana inicial

Segunda ventana

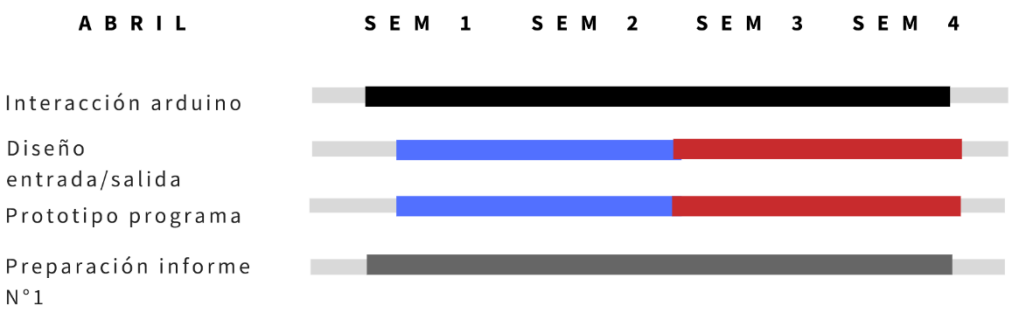
Datos a gestionar

El circuito Arduino posee un fotorresistor conectado a 2 resistencias auxiliares y una fuente de voltaje. El programa en Arduino mide el voltaje absorbido por el fotorresistor y entrega un entero binario dependiendo de la medición del voltaje (si el voltaje medido es mayor o igual a 3.3V, entonces el programa entrega un 1, de lo contrario, entrega un 0). Dicho valor es entregado al programa en Java como un número entero y es utilizado como tal. Además, cada fotorresistor es medido por un puerto específico, el cual es identificado por un número entero. El programa en Java debe recibir junto con las mediciones, el puerto desde el cual proviene, con el fin de guardar la medición más reciente en el espacio designado para cada puerto.

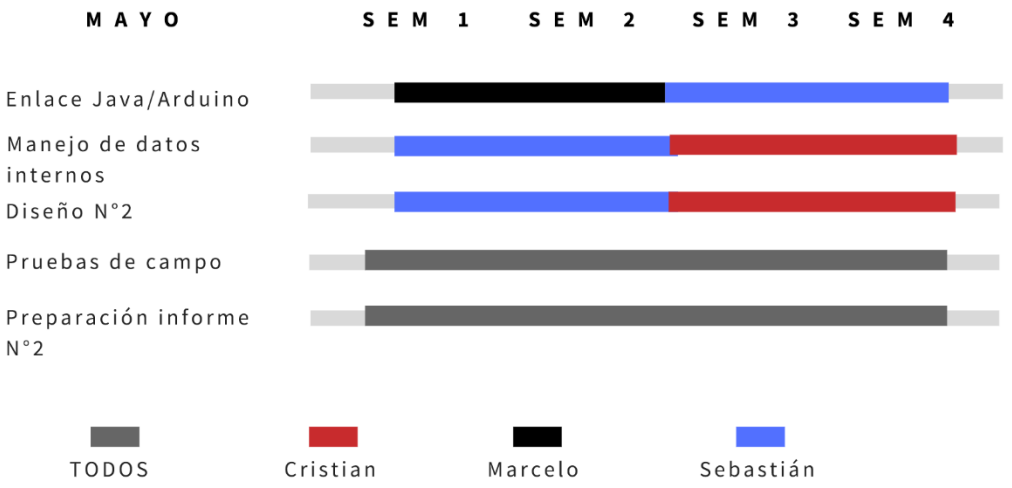
El programa no utiliza ninguna herramienta o archivo auxiliar para guardar sus datos, es decir, los datos sólo durante el tiempo de ejecución del programa.

Plan de Trabajo

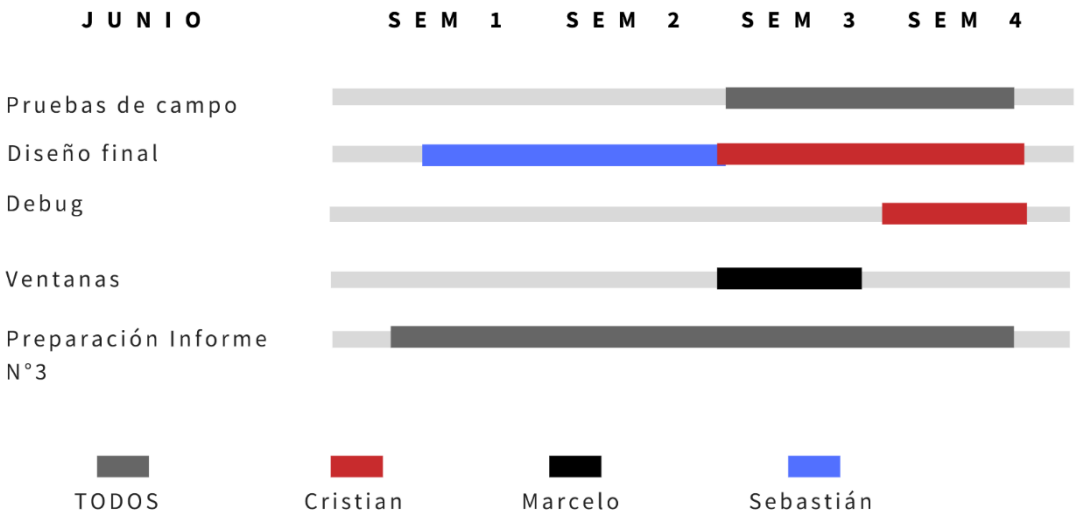
SMART PARKING



SMART PARKING



SMART PARKING



Conclusiones

Podemos concluir que es posible obtener información del medio mediante un circuito y programa Arduino. También es posible utilizar un programa en Java que interprete, procese y utilice dicha información. Sin embargo, todavía hemos de consolidar un método que nos permita enviar la información obtenida por Arduino a el programa en Java. Lo cuál será nuestra primera prioridad en la siguiente sección de este proyecto. Una vez logrado lo anterior, podremos centrarnos en la interfaz de usuario y optimización general del código.

Link al repositorio del proyecto:
https://github.com/cvidalse/proyectoprogramacion/tree/180857832a23c1a9fd470529f40b6885162045f3