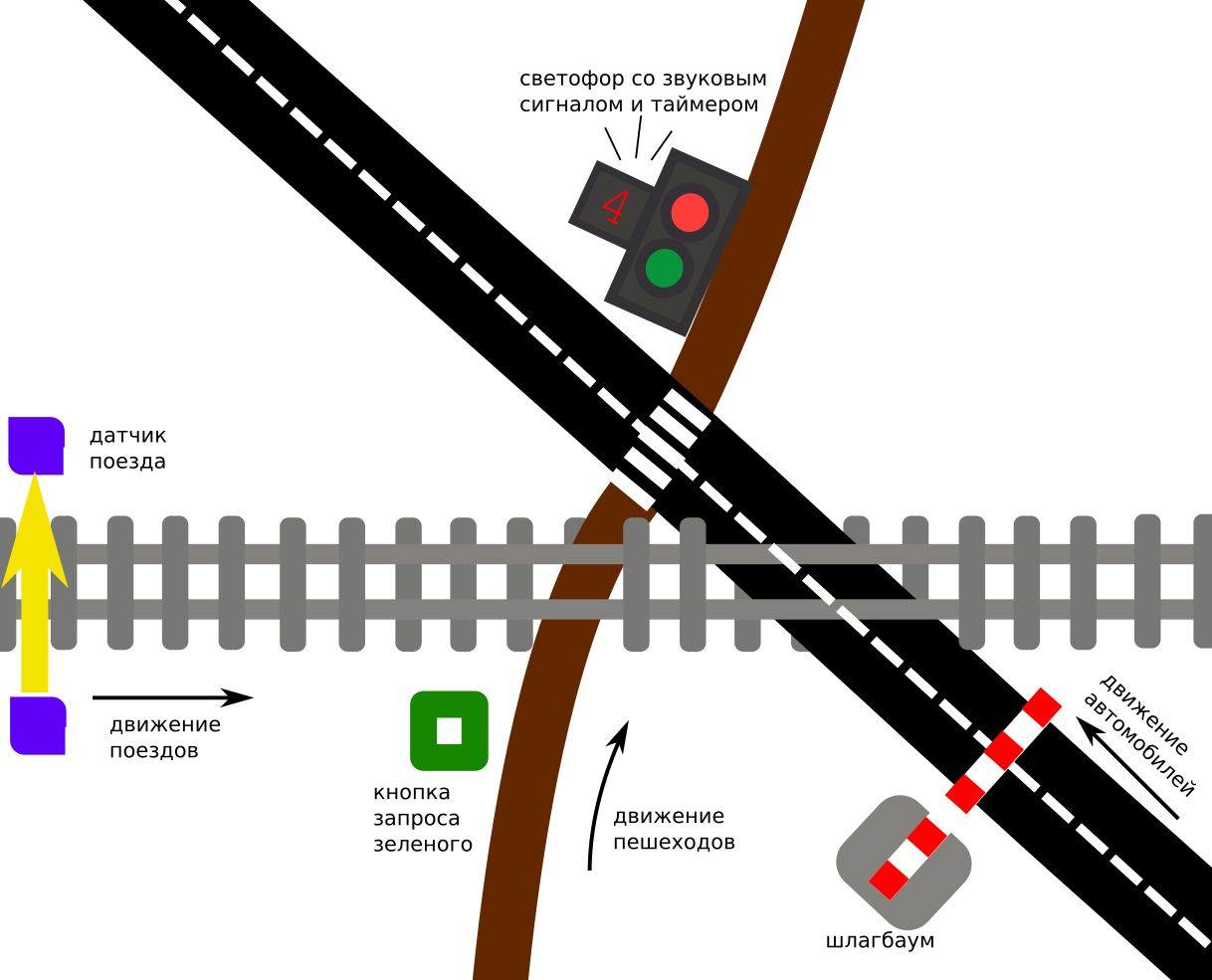
Hace un tiempo que venía pensando en hacer algo con Arduino que uniera el mundo físico con el de la programación. Comúnmente, al desarrollar solo interactuamos con las clásicas interfaces de usuario como el teclado o el mouse (y con suerte una pantalla táctil), pero Arduino me resulta interesante ya que permite obtener información de diferentes sensores y generar eventos a partir de ellos. Hay muchísimos sensores, como por ejemplo de temperatura, de luz, de presencia, de movimiento, de distancia, de inclinación, de gas, de fuego, de agua, pulsadores, etc.

Buscando información de cómo comenzar encontré que siempre se arranca con el mismo ejemplo: cuando se presione un botón pulsador se mantenga encendido un led por algún tiempo. Aunque esto parezca algo demasiado sencillo se puede extrapolar a diferentes conceptos, o sea, si reemplazamos el botón con un sensor como uno de luz y el led por una lamparita, podemos definir que si detecta que la luz de ambiente baja de un determinado umbral se prenda la luz. Otro ejemplo sería reemplazando el botón por un sensor de nivel de agua y el led por un motor centrífugo, podemos hacer que el motor suba agua a un tanque si el nivel del mismo está en un mínimo y apagarlo después de un tiempo.

Pasado el primer ejemplo comencé por hacer un curso en <https://www.coursera.org/> que brindaba la Universidad Autónoma de México, pero me pareció extremadamente básico (creo que el público objetivo eran chicos que apenas terminan la primaria). Luego encontré otro curso en la misma página, pero brindado por el Instituto de Física y Tecnología de Moscú el realmente me pareció muy bueno y una de las tareas es el proyecto que describo en esta nota.

**El desafío**

Este curso tenía como tarea crear un robot (en su definición más formal) el cual controlara un modelo que representara un cruce de las vías del tren con un camino peatonal y una vía para autos.



Por un lado, tenemos que detectar cuando se acerca el tren, en este caso se usó un sensor de distancia por ultrasonido, y cuando un peatón quiere cruzar para lo que use un botón de presión. Por otro lado, debemos informar a los vehículos que pueden o no pasar para los cual ponemos una barrera en el camino, de la misma forma tenemos que avisar a los peatones cuando pueden o no pasar para los cual use un semáforo, o sea un led verde y uno rojo, y la generación de un sonido, por medio de un buzzer.

El modelo debía cumplir con las siguientes reglas:

* Si aparece el tren, el semáforo se tiene que poner en rojo y bajar la barrera.
* Una vez que paso el tren debe mantener el semáforo en rojo y abrir la barrera.
* Si un peatón quiere cruzar debe presionar el botón el cual baja la barrera y pone el semáforo en verde.
* Si el peatón presiona el botón mientras pasa el tren, una vez que este termina de pasar el semáforo debe ponerse en verde y mantener la barrera baja.
* La luz verde del semáforo se mantendrá encendida durante 5 seg y después será intermitente por los próximos 4 seg. El sonido de la alarma deberá cambiar también al llegar a los 4 seg.
* El peatón deberá esperar 18 seg desde la última vez que se le dio paso hasta que se le permita nuevamente. Los primeros 9 seg tendrá la luz del semáforo en verde, los siguientes serán los vehículos quienes tengan habilitado el paso.

Al parecer estas reglas son sencillas pero fue realmente desafiante lograrlas.

**Comiendo al elefante**

En este caso los pequeños bocados fueron:

1. Planeación. Hacer un diagrama del mundo físico, la señal eléctrica y el código.
2. Diseño. Hacer un diagrama de estados
3. Construcción. Construir el código
4. Implementación. Construir el modelo físico

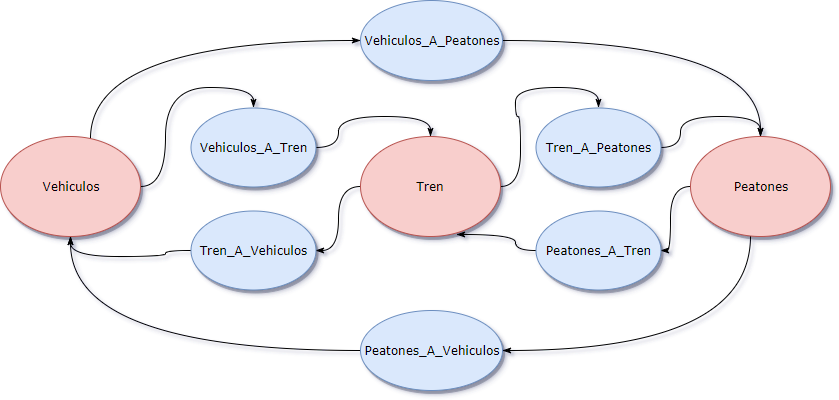
**Planeación**

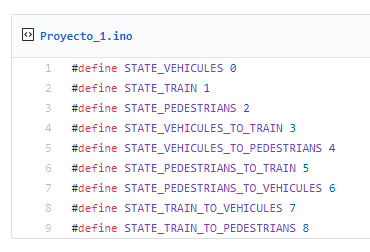
Sabía que tenía ciertos requerimientos que se iban a convertir en componentes específicos, como por ejemplo para el semáforo necesitaba dos led uno rojo y otro verde, esto a nivel de la señal eléctrica requiere de un pin de la placa Arduino para cada uno. Para detectar el tren necesitaba algún tipo de sensor de presencia y decidí usar un medidor de distancia por ultrasonido (podría haber usado uno por IR, o un sensor de luz, etc.). En este caso el sensor requiere 2 pines más. También necesitaba el botón pulsador para los peatones y un servo para subir y bajar la barrera que habilita el tránsito de los autos. Otros componentes son: el servo para manejar la barrera, el buzzer para la señal sonora del aviso del tren.

|  |  |  |  |
| --- | --- | --- | --- |
| Mundo Físico | Señal Eléctrica | Cant Pin | Código |
| Led Rojo | Digital | 2 | greenTrafficLight() |
| Led Verde | redTrafficLight() |
| Servo | PWM | 1 | openBarrier() |
| closeBarrier() |
| Press Button | Digital | 1 | digitalRead() |
| Ultrasonido | Analógico | 2 | measure() |
| Buzzer | Digital | 1 | beep() |
|  |  |  |  |
|  |

**Diseño**

Definí una máquina de estados para representar sus estados y sus transiciones ya que son muy marcados, por ejemplos no hay manera de que cuando este habilitado el paso del tren se habilite a los peatones o los vehículos. Por otro lado, las transiciones también son muy representativas, o sea, si estaba pasando el tren y fue presionado el botón de los peatones la transición es TREN\_A\_PEATONES con los cual es más fácil decidir mantener baja la barrera y poner el semáforo en verde.

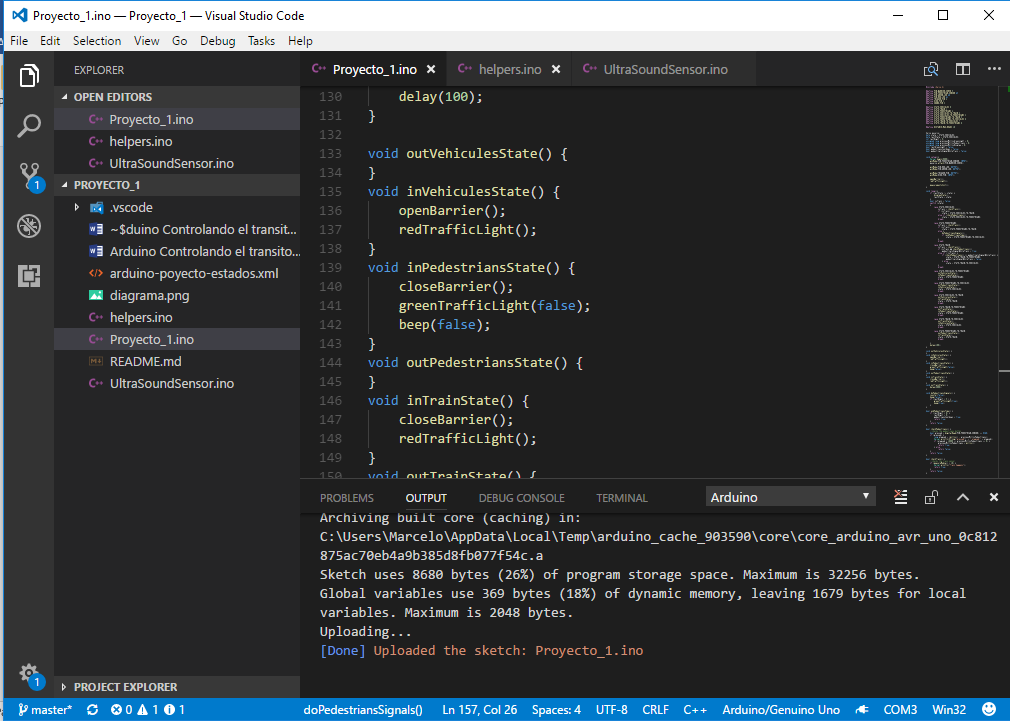




**Construcción**

Comencé usando el IDE propio de Arduino, pero realmente no es más que un notepad con los botones de validar y subir, por lo que casi lo descarte al momento. Buscando un IDE mejor para trabajar encontré un plugin para Visual Studio Code que integra con Arduino el cual me pareció bastante bueno. Este plugin permite tener en el IDE completador de código, descripción de las variables, ayuda sobre las funciones, poder engancharlo con el GIT, etc. Todas funcionalidades que no tiene el IDE propio de Arduino, por supuesto además de permitir validar (Ctrl-Alt-R) y subir (Ctrl-Alt-U) el código al Arduino, ver la consola del puerto serie y hacer un debug del programa (algo que no probé).

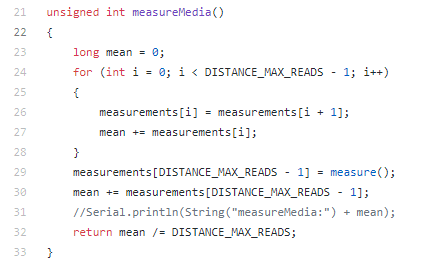
Una decisión de código fue separar en archivos diferentes ([helpers.ino](https://github.com/MarceloMosquera/ArduinoProyecto/blob/master/helpers.ino" \o "helpers.ino) y [UltraSoundSensor.ino](https://github.com/MarceloMosquera/ArduinoProyecto/blob/master/UltraSoundSensor.ino)) las funciones que no son específicas de este proyecto, por ejemplo el que hace la medición del sensor o el que sube o baja la barrera, dejando en el archivo principal ([Proyecto\_1.ino](https://github.com/MarceloMosquera/ArduinoProyecto/blob/master/Proyecto_1.ino)) las funciones que tienen lógica de negocio.

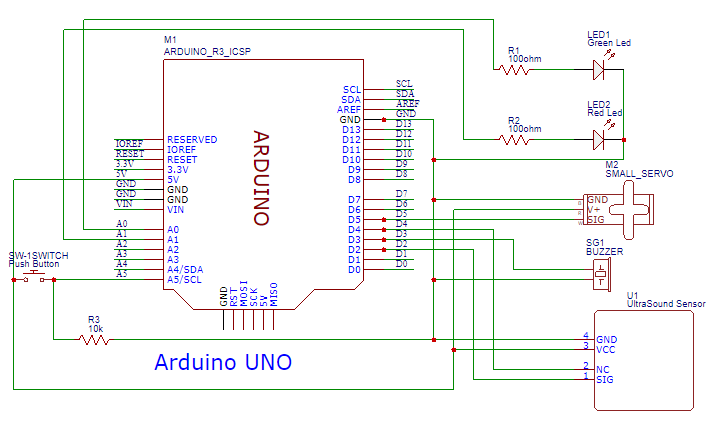


**Implementación**

Arranque por poner los leds y el botón para generar el cambio de estado para habilitar a los peatones. Después puse la barrera y el estado de los vehículos y por último el sensor del tren con sus cambios de estados y transiciones.

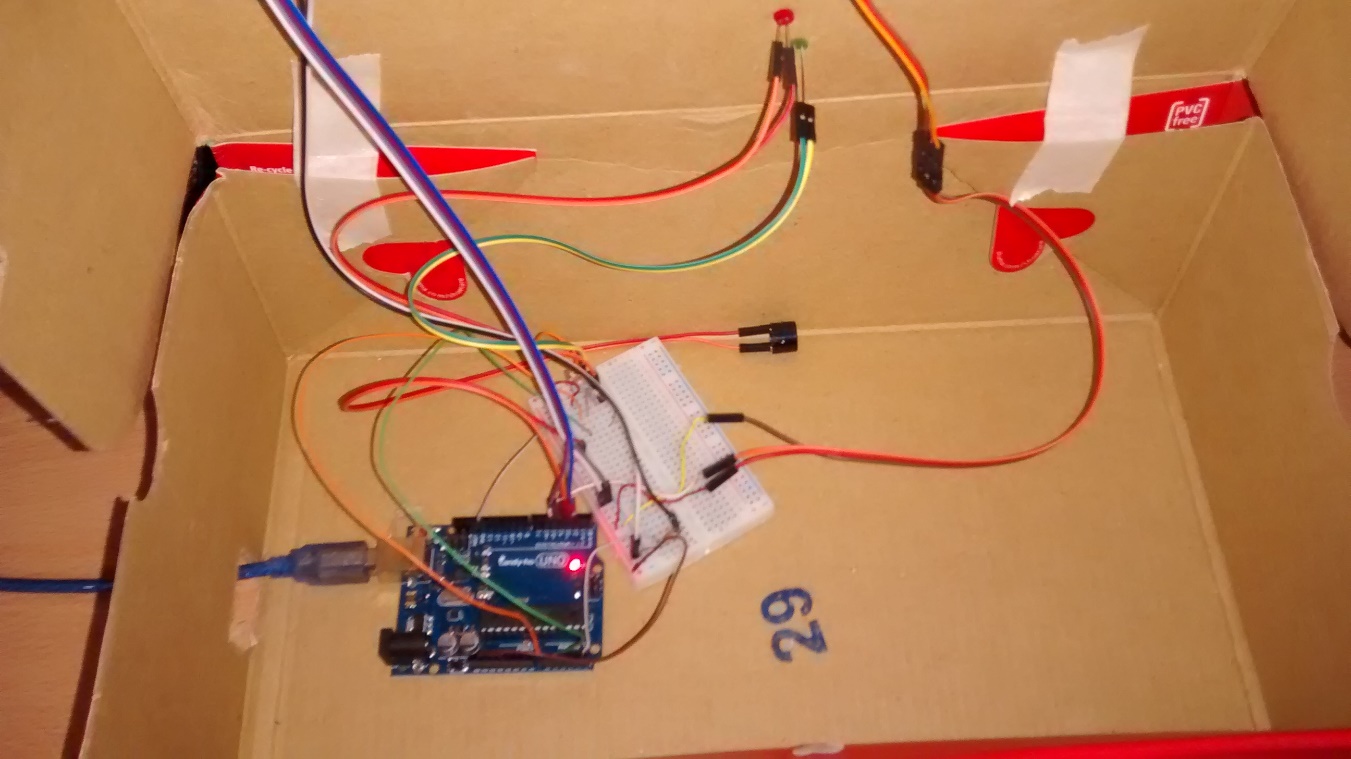
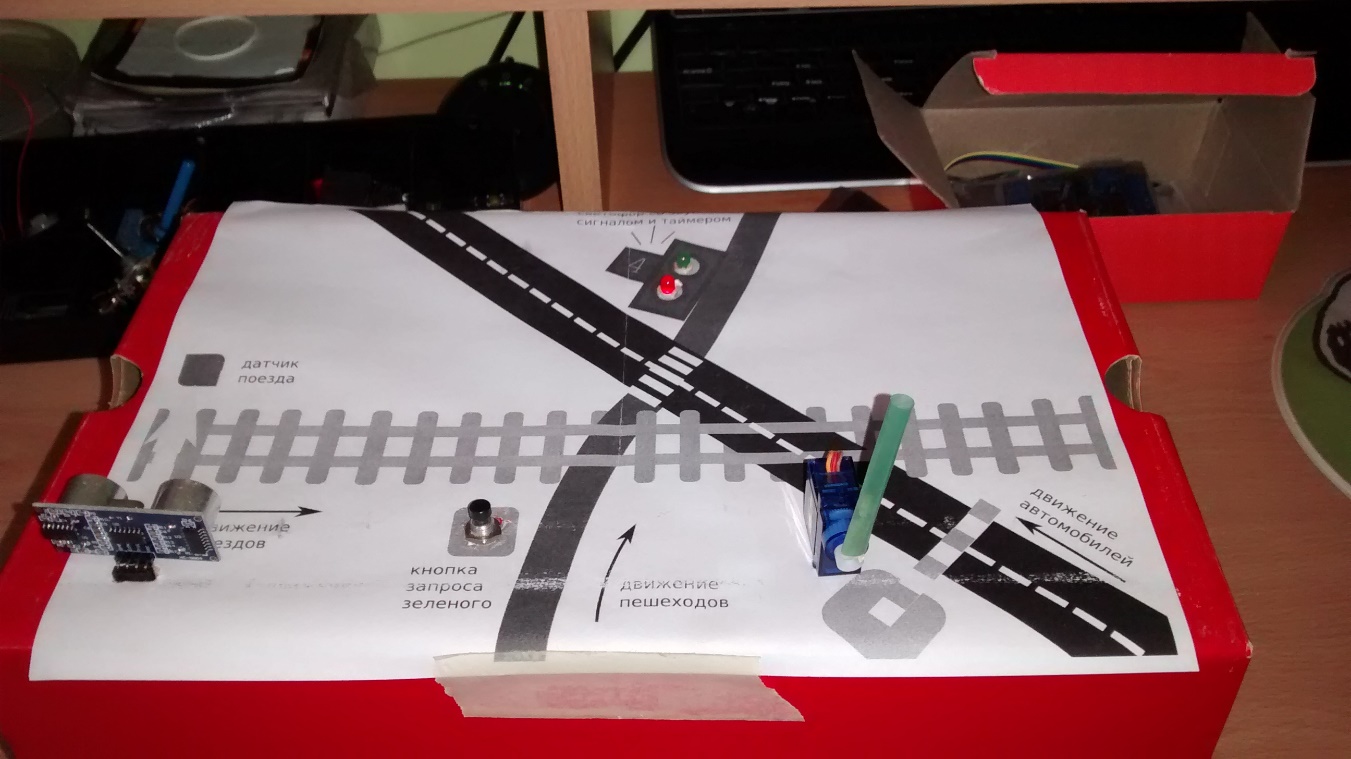
En ese momento me di cuenta que el sensor a veces tiraba un valor muy bajo y que por lo que llegué a leer puede deberse a la baja calidad de los componentes, la interferencia en los cables o los cambios de voltajes generados posiblemente por el servo de la barrera. Para solucionarlo redefiní el método que mide la distancia para que devuelva el promedio entre 10 lecturas, de esa manera se filtran los valores extremos.





**Conclusión**

Arduino permite unir el resultado de varios sensores y generar eventos en los componentes de salida de una forma sencilla y a muy bajo costo. Obviamente hay que tener ciertos conocimientos de electrónica y de programación básicos, pero la experiencia se puede ir adquiriendo a medida que uno construye.



**LINKS**

* Repositorio del proyecto: <https://github.com/MarceloMosquera/ArduinoProyecto>
* Curso: https://www.coursera.org/learn/arduino/home/welcome
* TinkerCad: https://www.tinkercad.com/things/6sMedEaTgCl-proyecto