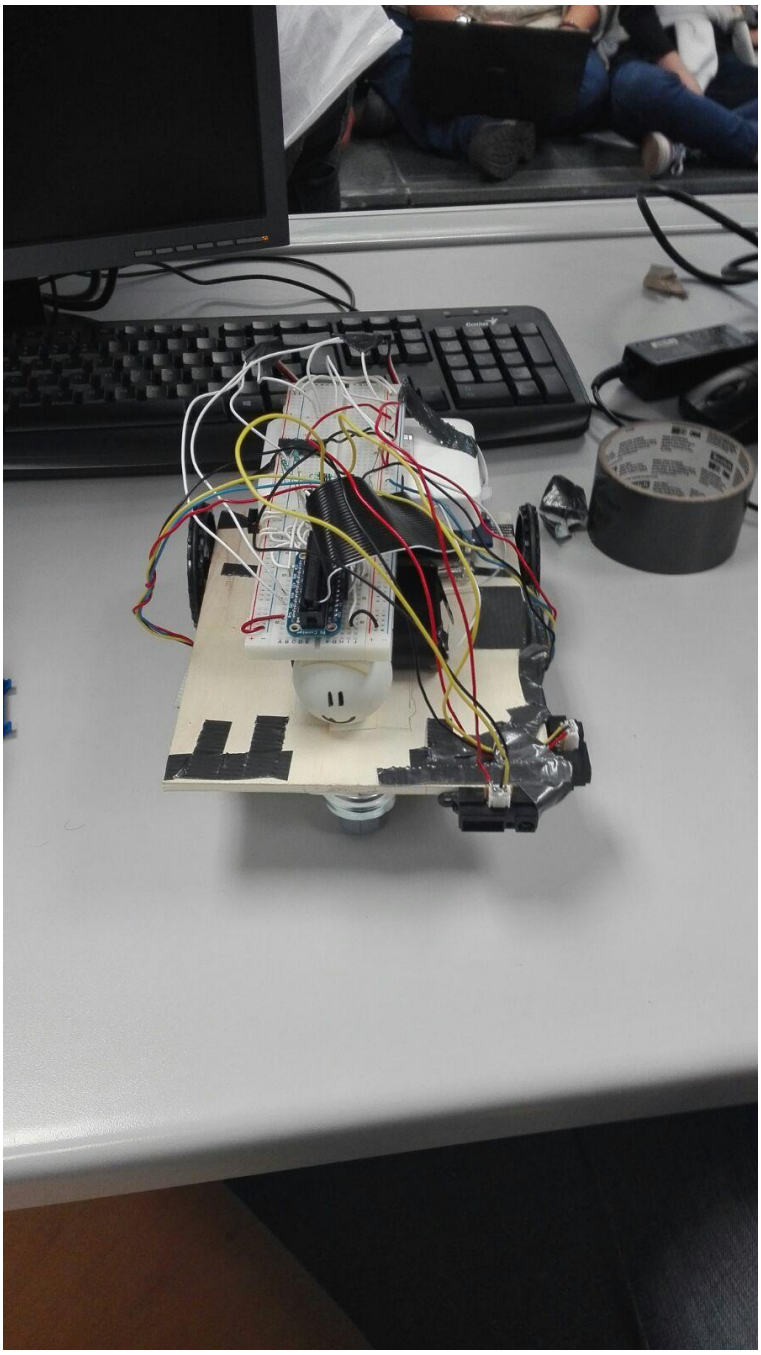


Teseo

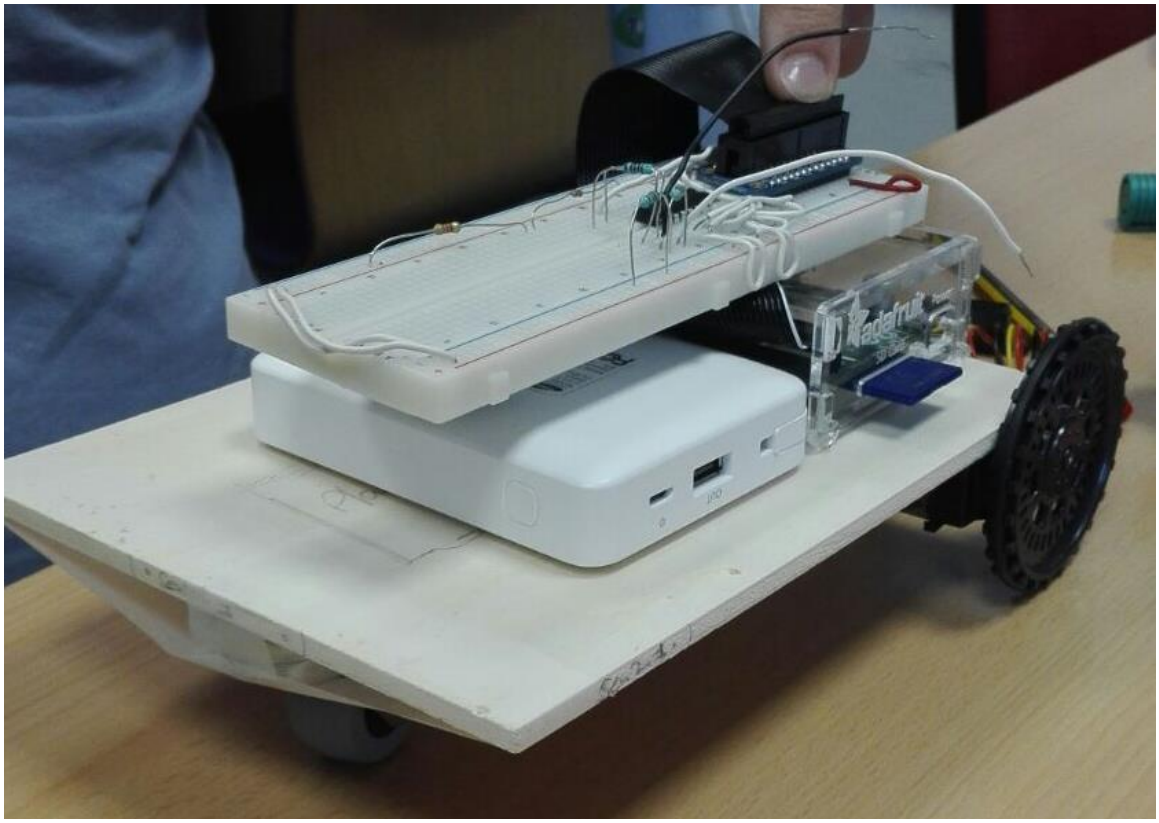
PROYECTO DE ROBÓTICA 2016-2017

Grupo 5 | Robótica | 19/01/2017

Introducción y Montaje

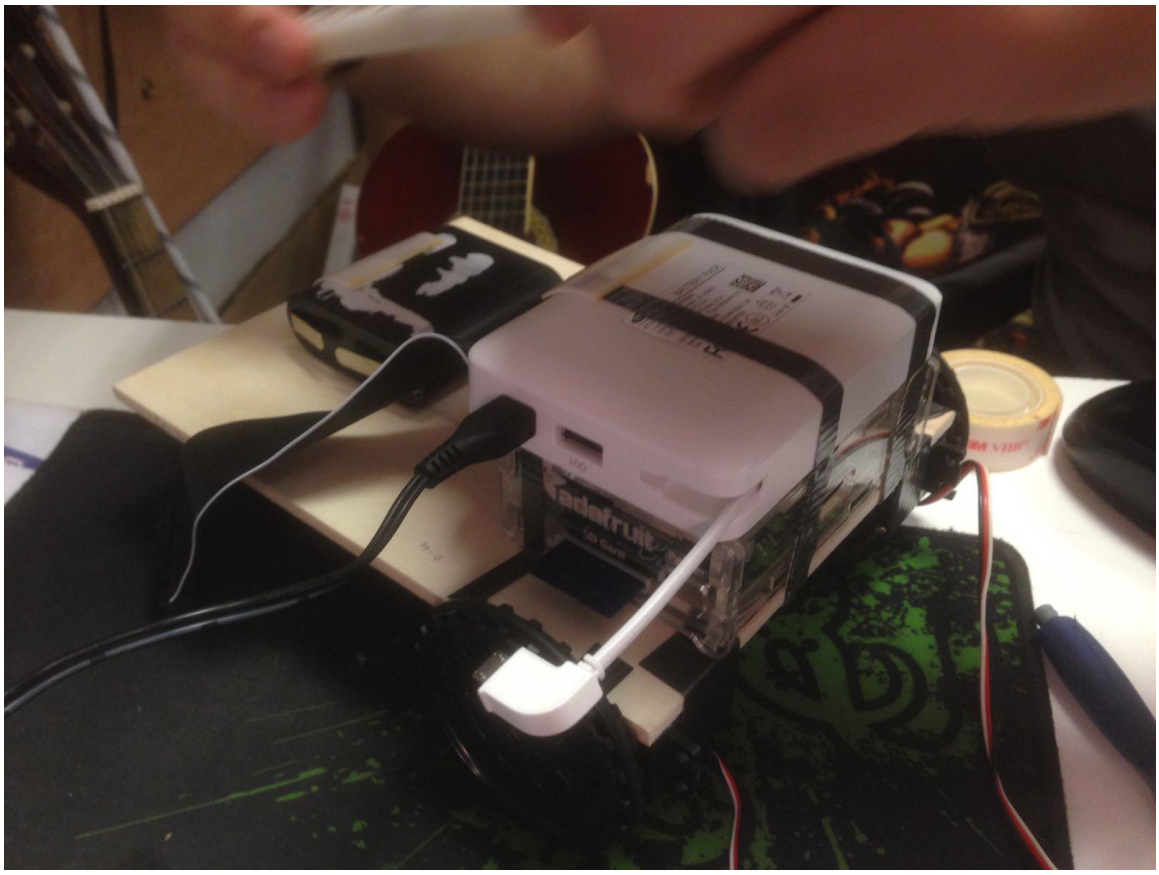
Teseo entró en el laberinto del Minotauro, se encontró con este, y le dio muerte a puñetazos.

Para el montaje del robot, utilizamos una plancha de contrachapado para poder sujetar los componentes, una rueda de mueble convenientemente engrasada, y una pila de 4.5V, y una batería portátil para móviles aparte del material proporcionado.

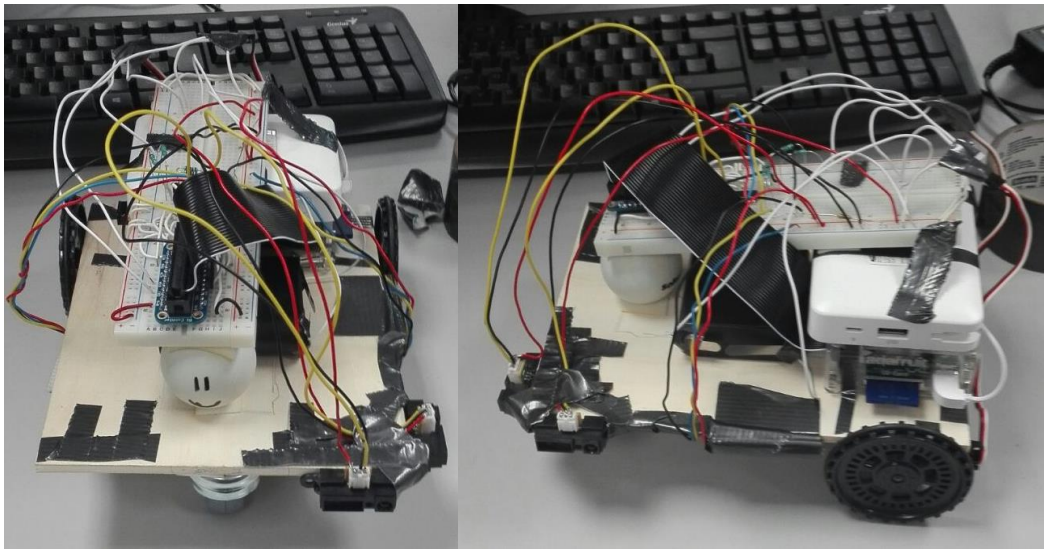


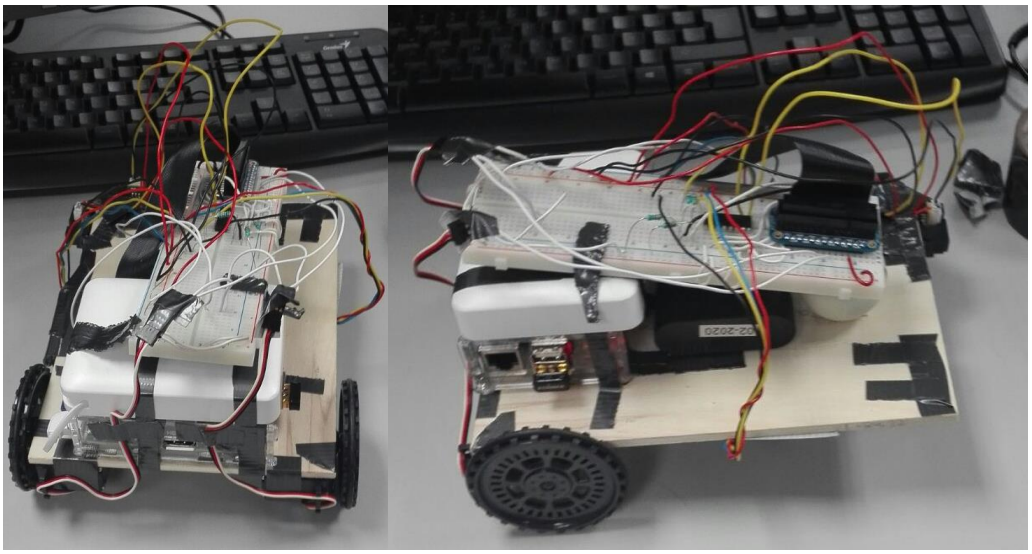
En primer lugar, optamos por colocar los componentes de una forma 'ergonómica'. No encontrábamos una forma cómoda para colocar la pila y alimentar a la raspberry con la batería, así que nos decantamos por un modelo en 'puente' en el que utilizamos una pelota de pingpong para sujetarlo todo (Así en frio no parece tan buena idea).

Después, montamos el robot, mientras programamos las rutinas, así que no estábamos seguro de las posiciones de los sensores o de los servos, o incluso de la raspberry + batería, así que para mantenerlo todo utilizamos cinta adhesiva de doble cara, y bastante cinta americana (cinta de ductos).



El producto final, tras varias pruebas de software, fue de la siguiente manera:





De esta manera, la pila de 4.5V se puede cambiar de forma sencilla (En la entrega íbamos por la tercera), y en la tercera y cuarta imagen se puede ver que los puertos USB y HDMI eran accesibles para programar.

Los sensores infrarrojos CNY70 van por debajo y en el centro de la placa. Al tener nuestro robot tracción trasera, tenerlos delante del todo habría provocado problemas al seguir una línea con una curva muy cerrada.

Los sensores de proximidad GP2Y0A41SK0F los montamos casi juntos, uno en el lado izquierdo y por la parte delantera, casi juntos. De esta manera no tendríamos puntos muertos con una posible esquina (Originalmente el delantero iba en el centro, pero gracias a la magia de la cinta aislante lo pudimos cambiar).

PROGRAMACIÓN

Para programar, montamos una raspberry Pi model B accesible por SSH con la librería de wiringPi instalada para hacer pruebas, que subíamos vía Git para poder probar y ver si compilaba.

Utilizamos tal cual el código del sensor de proximidad, y definimos un montón de constantes para poder ajustar cosas sobre la marcha:

- GPIOS de los servos (LENGINE_GPIO, RENGINE_GPIO).
- Señales para avanzar, retroceder o para en cada servo (LENGINE_FORWARD_FAST, RENGINE_BACKWARDS_FAST, etc.).
- GPIOS de los sensores infrarrojos (LS_GPIO, RS_GPIO).
- GPIOS de los sensores de proximidad (SENSOR_CENTER, SENSOR_SIDE).
- Constantes para la inicialización de los sensores de proximidad.

Añadimos constantes para la navegación a modo de números mágicos (no hay otra forma), que, aunque al principio eran más, terminaron siendo 4:

- SENSOR_ACCEPTABLE_MIN, que define la distancia máxima aceptable a la que tenemos que tener la pared izquierda: 350
- SENSOR_ACCEPTABLE_MAX, que define la distancia mínima aceptable a la que nos podemos aceptar de la pared derecha: 450.

El rango es de aproximadamente 100, que no es en sí mucho, pero jugando con rangos y valores decidimos que era la mejor distancia para poder ejecutar giros en cualquier dirección sin atascarse.

-SENSOR_CENTER_TURN_DISTANCE, que define la distancia a la cual empezamos a hacer un giro hacia la derecha al encontrar un obstáculo delante nuestro (y sigamos teniendo pared a la izquierda, ergo, giro a la derecha): 300

-SENSOR_CENTER_COLLISION_DISTANCE, que define la distancia frontal a la cual, si hacemos un giro cerrado, no colisionaremos de frente. La utilizamos sobre todo cuando en un giro hacia la derecha nos encontramos un obstáculo de frente, como, por ejemplo, un giro de 180º demasiado cerrado: 650

Con estos dos últimos valores podíamos hacer giros de cualquier índole sin ningún problema.

Las dos rutinas posibles de Teseo son llamadas recursivamente hasta que llegue la señal de interrupción.

Por último, decir que el programa, Theseus.c (que compilamos como Theseus) tiene parámetros de entrada:

- -f: Sigue una línea (follow)
- -s: Soluciona un laberinto (solve)

El código se halla comentado, tiene la licencia MIT y está publicado en GitHub, junto con el resto de las prácticas, aquí.