Universidad del Valle de Guatemala
Algoritmos Y Estructura de Datos
Sección 10
Grupo 6

DELVALLE
GRUPO EDUCATIVO

Dieter de Wit 15146
Robbin Woods 15201
Gerardo Cardoza 15410

Proyecto # 1: Fase 1

Algoritmos para la Resolución de Laberintos:

- A continuación, se encontrarán algunos algoritmos eficientes para la resolución de laberintos que encontramos:
    - Wall Follower (Mano Derecha): Este algoritmo funciona en cualquier laberinto de esquinas, laberintos cuadrados. No utiliza memoria adicional debido a que utiliza un concepto humano, si se recorre una pared con la mano derecha en ella y caminando hacia el frente en algún punto del laberinto se encontrara la salida. Cruce generalmente es siempre a la derecha. El método no necesariamente encontrara la solución más corta.
    - Pledge (Promesa): Se asigna el movimiento en una sola dirección o instrucción al robot hasta que este topa con una pared. Una vez lo hace se le genera otro movimiento, la confianza en que el movimiento será el que resuelva el laberinto es la clave para resolverlo. El algoritmo es ineficiente debido a que puede regresar al inicio o repetir algunos pasajes más de una vez.
    - Chain (Cadena): Resuelve en laberinto utilizando espacio de memoria. Genera fracciones de laberinto conforme se pasa por secciones más pequeñas y las resuelve en secuencia. Se especifica el inicio y la ubicación final deseada, el algoritmo siempre encontrara una solución en un tiempo corto, pero se debe de conocer las coordenadas del final, lo cual lo puede hacer no viable.
    - Tremaux: Este se centra en la solución de un algoritmo por parte de un humano. El método de marcar las partes del laberinto que ya pasaste con un cordón se sostiene guardando en memoria los lugares que ya paso el robot.
    - Dead End Filler (Maca Callejones): Se centra en estudiar el laberinto. Marca todos los puntos muertos donde el robo no pudo pasar el laberinto, luego lo último que queda tiene que ser la solución. Se usa normalmente cuando se desea recorrer el mismo algoritmo más de una vez en un tiempo considerablemente menor.

Pseudocodigo:

```
If (noParedIzquirda)
        cruce 90 grados izquierda;
elseif (noParedEnfrente)
        avanzar un espacio;
else
        cruce 90 grados derecha;
```

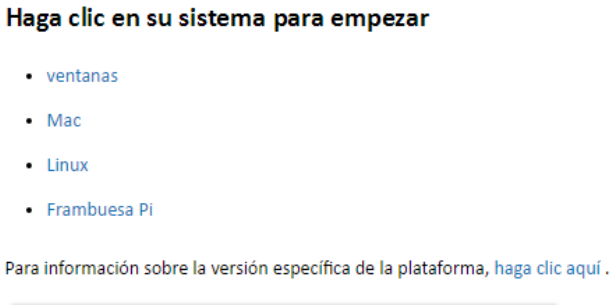
¿Por qué escogimos el Algoritmo?

- Elegimos el algoritmo de la mano derecha debido a que sale de cualquier laberinto cuadrado sin importar que tan difícil sea el mismo, se puede tardar un poco más pero su tiempo siempre será el mismo o similar. El algoritmo es fácil de implementar ya que no tiene tanta complicación en la forma del código.

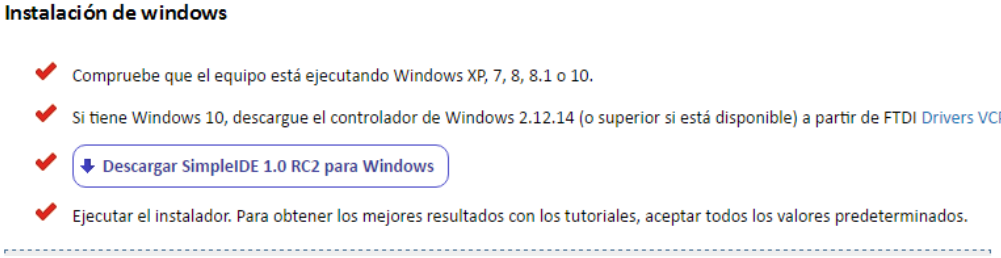
Proyecto # 1: Fase 2

Instructivo para instalar el simpleIDE:

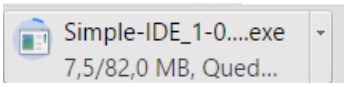
1. Para instalar el simple IDE, primero hay que buscar simple IDE en google.
2. Meterse a la primera página la cual es: http://learn.parallax.com/tutorials/language/propeller-c/propeller-c-set-simpleide.
3. Luego meterse en el sistema que corresponde.



   En este caso será Windows.
4. Luego darle en Descargar.

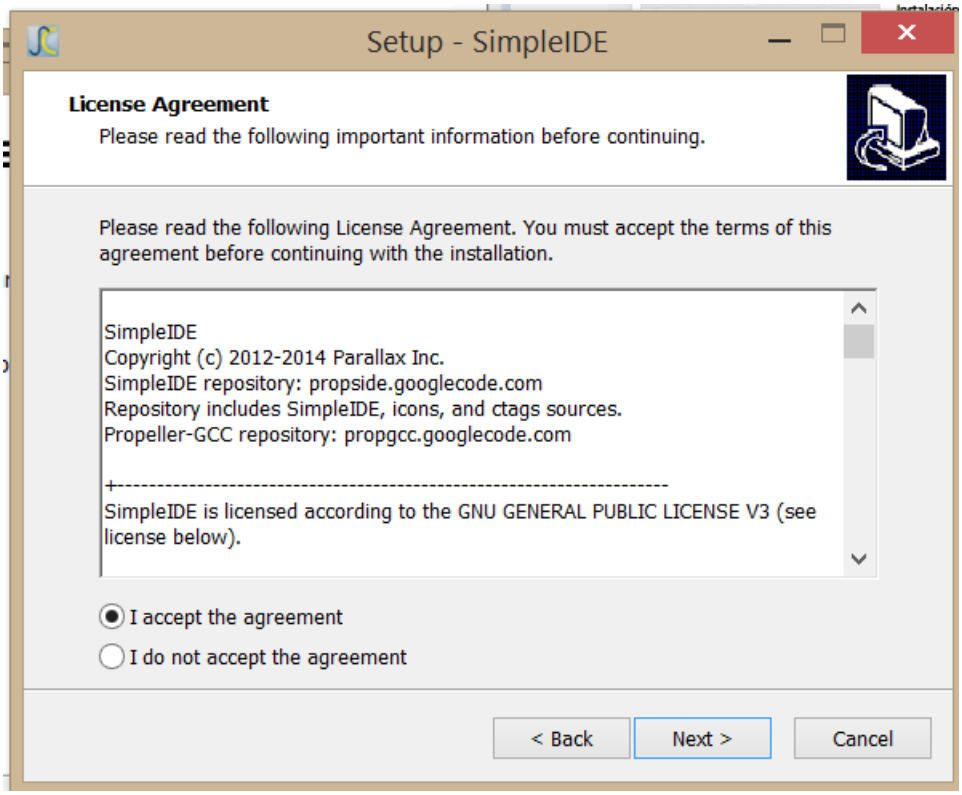


5. Esperar a que cargue para poder instalarlo.

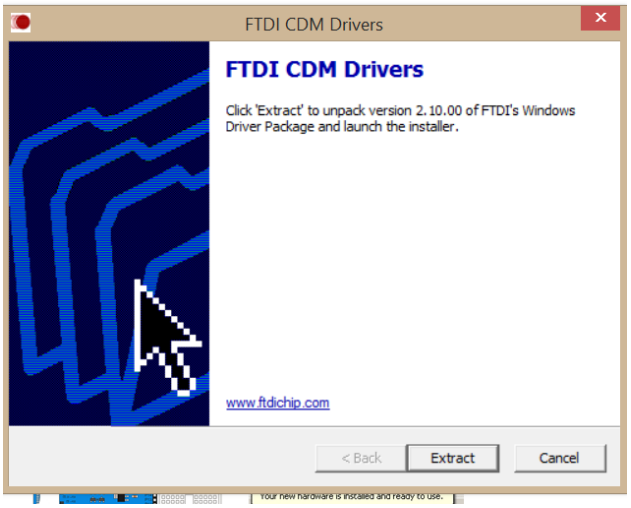


6. Después de lograr descargarlo aparece esto:

Darle next y luego aceptar los acuerdos que tiene el programa.



7. Darle extraer



8. Esperar y darle siguiente a lo otro y finalizar.

Activity BOT:

El robot en si cuenta con un multi-núcleo cerebral hélice microcontrolador con hardware.
Esté cuenta con:

- Hélice Versátil con un robusto chasis de aluminio.
- Servos de alta velocidad.
- Codificadores ópticos.
- Ruedas con neumáticos tóricas.
- Tarjeta SD, archivo de almacenamiento.
- Sistemas de navegación utilizando el tacto, la luz visible, luz infrarroja y sensores ultrasónicos.

Armado de conexiones eléctricas:

- 2 resistencias de 20 k ohmios resistencias.
- 5 pilas 1.5 AA pilas.

El uso de las resistencias se conecta P14 y P15. Y meter las pilas dentro del robot para que funcione.

Ultrasonido:

Este sensor permite detectar los obstáculos y medir la distancia a ellos. El microcontrolador Propeller puede marcar el tiempo que transcurre el programa activado. Retorno de eco puede llegar a utilizar para calcular la distancia a un objeto.

Infrarrojos:

Esta no es visible al ojo humano, pero el receptor de los infrarrojos detecta el patrón de luz intermitente. Recibe los datos de reflexión a partir de los receptores de infrarrojos y la utiliza para tomar decisiones de navegación.

Desempeño de Estructura de Datos:

- Algoritmo Utilizado: Mano Derecha.
- Estructura de datos: La clase que tiene las funciones de movimiento, funcionesMovimeinto.c se asimila a una clase con datos abstractos debido a que cualquier robot paralax podrá utilizar los movimientos de las mismas para cualquier algoritmo que se escoja.
  Además, se utilizaron colas debido a que cada vez que cada vez que se realiza alguna acción detiene las demás y luego continua hasta que termina la tarea.
- La comparación con el algoritmo anterior es que en este caso se tienen sensores que detectan la pared física, el algoritmo funciona igual sin embargo se tiene imperfecciones a la hora de realizarlo con el robot. Los giros no son exactos, los sensores presentan lecturas erróneas en algunos puntos del laberinto y hay laberintos donde la mano derecha puede no funcionar. El desempeño físico es más variable sin embargo se puede visualizar mejor cualquier error u imperfección.