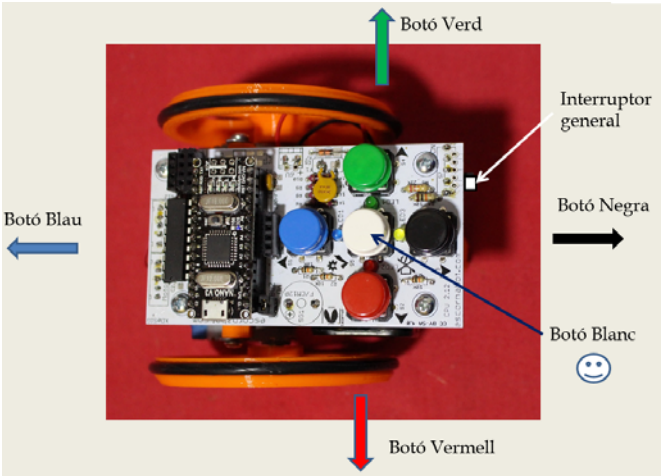
Entendre el Programa de Funcionament d'Escornabot

Introducció

Escornabot pot executar seqüencies de moviments que son programats pel usuari mitjançant la pulsació de les tecles del robot. També es pot ampliar el seu control amb un telèfon mòbil o una "tablet" a través de Bluetooth o Wifi.

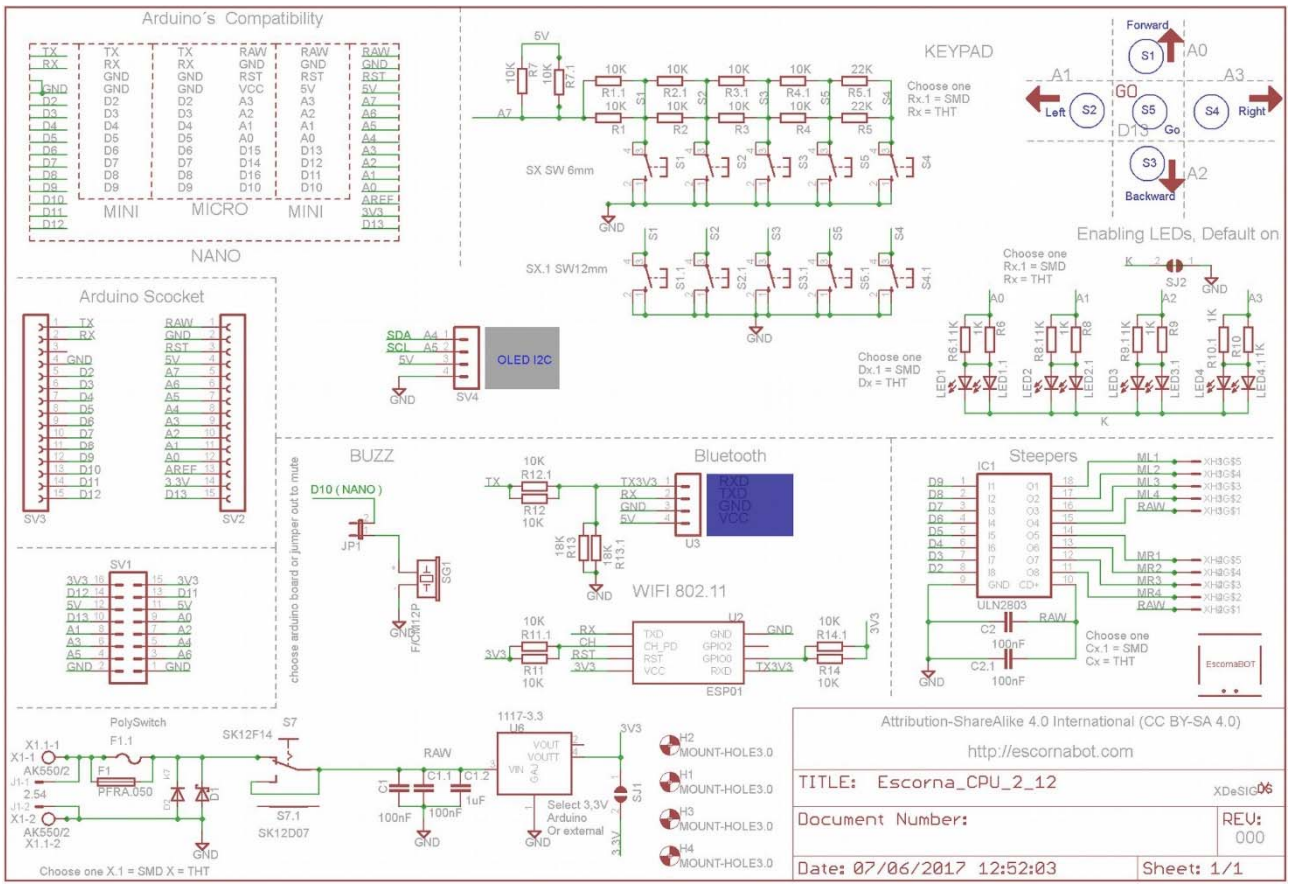
L'Escornabot disposa d'un interruptor general de posta en marxa i 4 botons que permet programar els moviments que volem que faci, cada pulsació programada farà avançar 10 cm, permet moure endavant (tecla blava), moure enrere (tecla negre), girar a l'esquerra (tecla vermella), girar a la dreta (tecla verda) i la tecla del mig de color blanc farà que executi la seqüencia programada. També la tecla blanca pot parar la seqüencia quan l'Escornabot està en marxa.

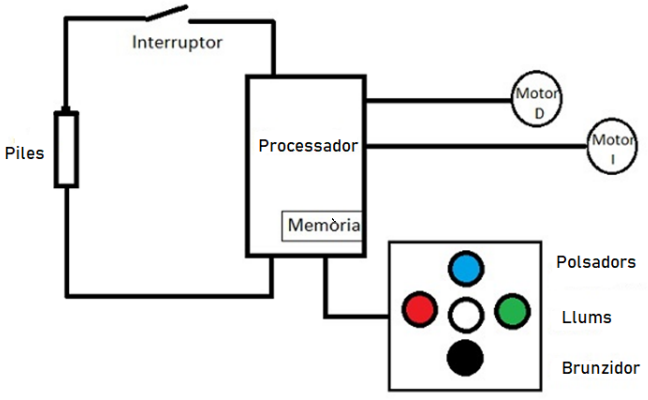


Components de l'Escornabot versió Singularis

- ⊡ 2 motors per moure cada roda
- ⊡ 1 processador per la seqüencia de les ordres cap els motores i LEDs de senyalització.
- ⊡ 1 memòria per guardar les ordres.
- ⊡ 5 tecles per polsar les ordres.
- ⊡ 5 llums d'indicació.
- ⊡ Brunzidor per sentir que està fent.
- ⊡ 4 piles per donar energia a l'Escornabot.
- ⊡ Interruptor per posar para posar en marxa.

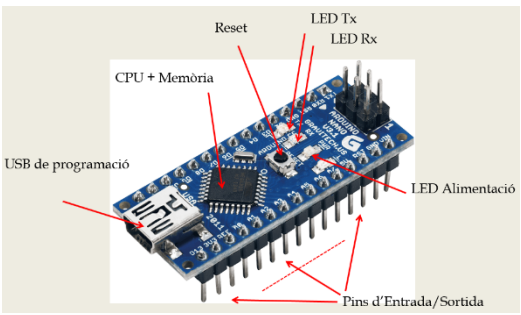
Esquema de l'Escornabot Singularis



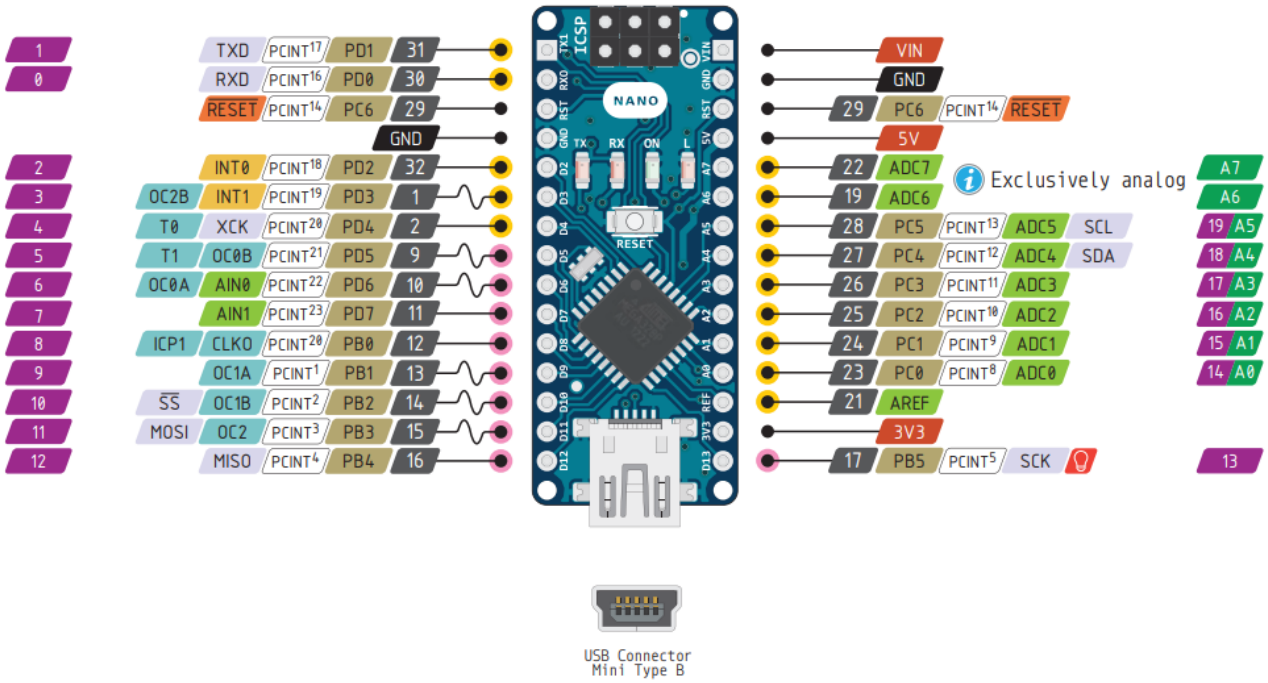


Processador Arduino

La unitat de control de procés Arduino es el cervell del robot, es on es guarden les instruccions rebudes des del teclat i on es controlen els motors, LEDs i brunzidor. També incorpora un connector USB per poder actualitzar el firmware del robot.



Aquest placa de circuit imprès d'Arduino (anomenada Arduino Nano) conté un microcontrolador de 8 bit del fabricant Microchip de la família AVR (abans Atmel) ATMEGA328P, incorpora en el seu interior la unitat central de procés (CPU), una memòria Flash de 32KBytes (on hi ha el programa y es pot reprogramar), una memòria RAM 2 KBytes (per emmagatzemar valors temporalment, ja que quan es treu l'alimentació es perden aquestes dades i una memòria Eeprom 1 KBytes (on es guarden dades durant l'execució d'un programa i romandran encara que es tregui l'alimentació).



Entrades analògiques van al convertidor analògic digital (ADC de 10 bit) i les sortides analògiques poden ser PWM (pera el control de motores).

Entrades/sortides digitals i altres permeten una comunicació sèrie (I2C, SPI y UART). Un pin de Reset que va al polsador de la placa.

Una part imprescindible es un rellotge de quars de 16 MHz amb un oscil·lador fins 20MHz que equival a poder executar 20 milions de instruccions per segon.

Una anotació més sobre la memòria Flash, es on es programa un firmware anomenat Bootloader (que ocupa 0,5 KBytes), permet carregar el programa que podem fer via sèrie a través del USB.

Aquesta placa s'alimenta amb 5V

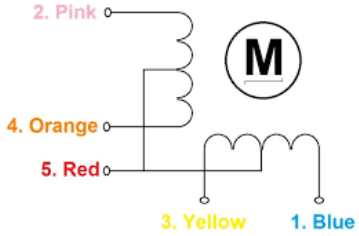
Motor pas a pas

L'Escornabot incorpora dos motors pas a pas, vol dir que es mouen per impulsos, dintre del motor incorporen una reductora, cada 64 polsos dona una volta, que permet una gran precisió. Segons la freqüència dels polsos proporcionen la velocitat.

Si giren en el mateix sentit el robot avança o retrocedeix, si un gira en un sentit i l'altre en sentit contrari, el robot girarà 90 graus cap a la dreta o 90 graus cap a l'esquerra.

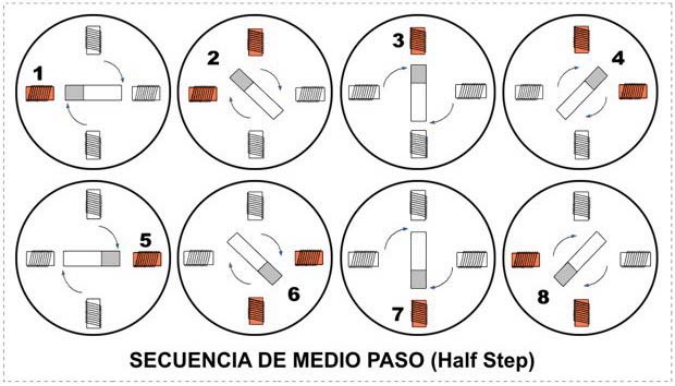
Hi ha una nova revisió del firmware, la 1.6, que pot girar a 60 o a 90 graus, aquest es per usuaris més avançats al permetre anar en diagonal.

Aquest tipus de motor pas a pas es de 4 fases Unipolar i s'utilitza la commutació de Mig Pas.



Secuencia de conmutación de Medio Paso

Número y color de pata	→ Dirección de la agujas del reloj (fases 1-2)							
	1	2	3	4	5	6	7	8
(4) Naranja	■	■						■
(3) Amarillo		■	■	■				
(2) Rosa				■	■	■		
(1) Azul						■	■	■



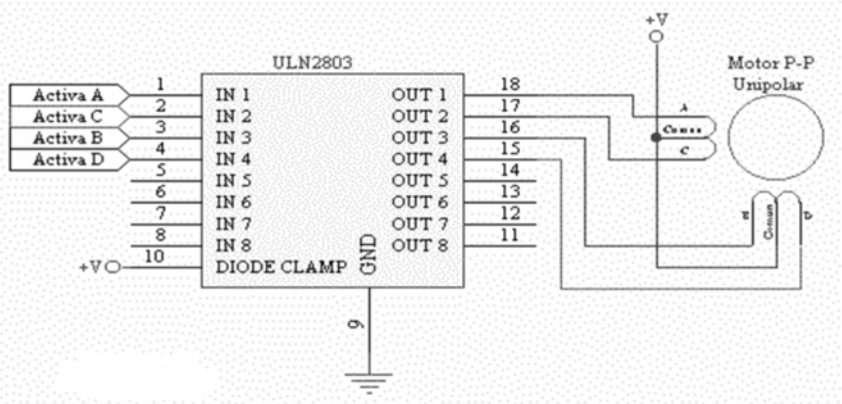
En aquesta configuració de control anomenada de Mig Pas, les dues bobines que tenen un punt mig formant quatre bobines, es poden veure com es van alternant los polsos en la taula anterior per realitzar els moviments pas a pas. Es necessiten 4 passos, un gir en vuit cicles i un gir complert de l'eix exterior 64 voltes del rotor, per el que es requereixen 2048 passos per una volta.

$$\text{Nº de passos per una volta} = 4 \times 8 \times 64 = 2048$$

Amb el següent codi aconseguirem una volta completa del motor:

```
const static uint8_t step_pattern[] = {
            B00001, B00011, B00010, B00110, B00100, B01100, B01000, B01001
                                                    } ;
```

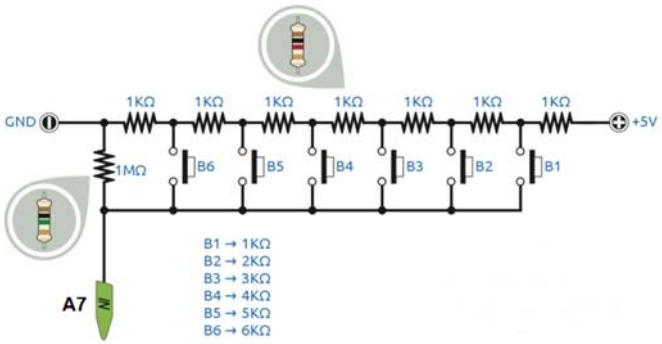
Es pot constatar que el patró del codi correspon a la taula de seqüencia de commutació de Mig Pas, cada 1 en la taula apareix un punt en vermell.



El corrent de cada bobina es relativament alta i es necessari posar un "driver" per que el microcontrolador no entregui aquest corrent si no que la entrega cada transistor intern.

Control del teclat

El teclat està minimitzat per només fer servir un sols pin, utilitzant una entrada analògica de la següent manera.

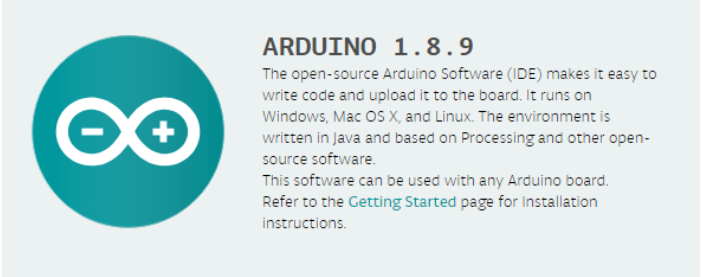


El convertidor analògic/digital del microcontrolador "llegeix" el valor de voltatge del pin d'entrada A7 i "entén" quin botó s'ha polsat.

Cóm es programa?

Escornabot fa servir un mòdul d'Arduino, molt popular en el mon DIY, on es grava el firmware de funcionament, que està totalment disponible i documentat en https://github.com/escornabot

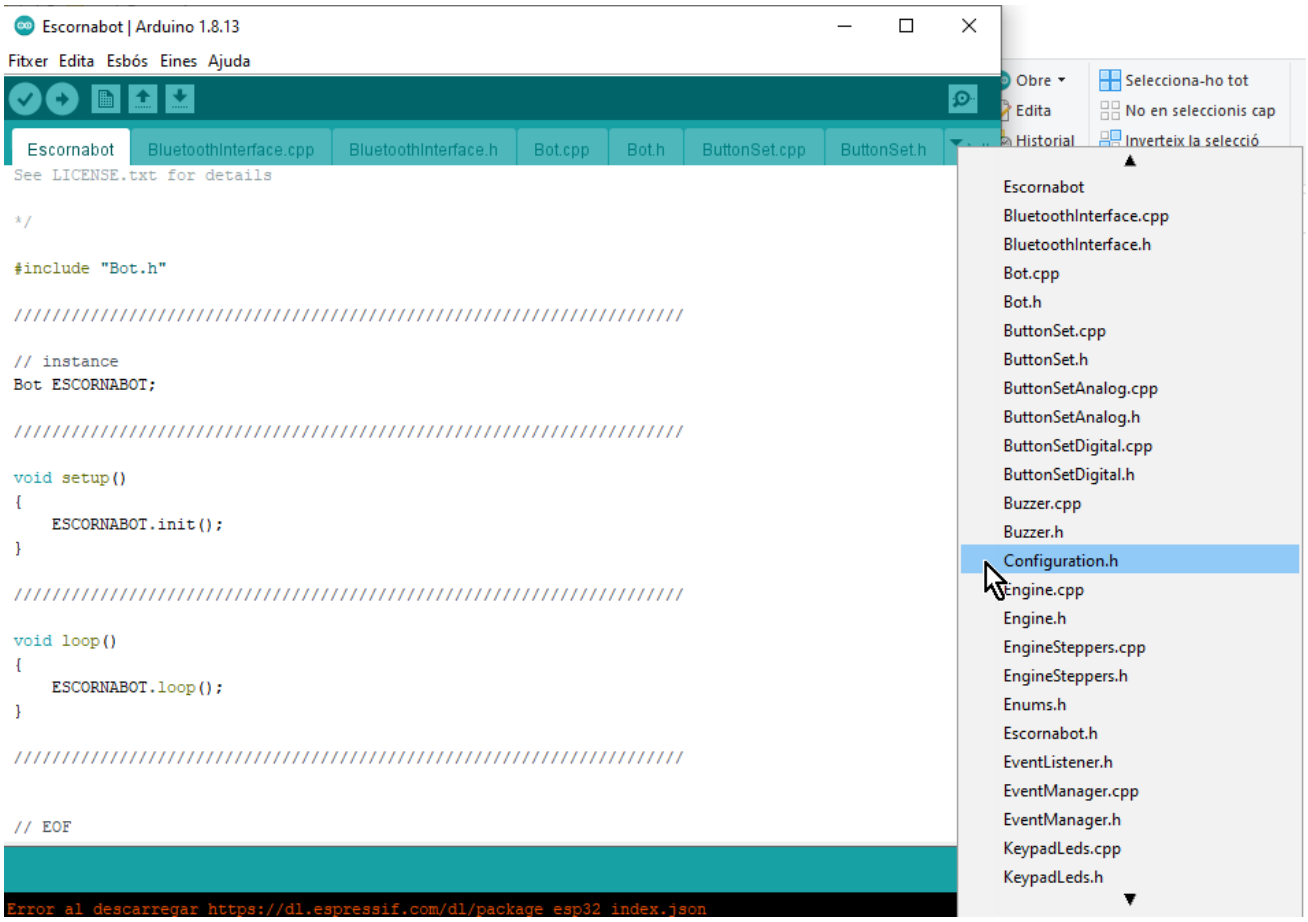
Es fa servir l'entorn de desenvolupament d'Arduino i es pot fer com un taller de informàtica el desenvolupament complet o parcial del seu funcionament, evidentment adequant-lo a l'edat corresponent.



Entendre el programa de funcionament

El programa d'aquest Escornabot ha estat realitzat per @caligari, seguidament es descriurà la part principal de funcionament.

Al obrir el firmware 1.4.3 o 1.6.2 Escornabot.ino amb l'entorn d'Arduino, apareixen una sèrie de pestanyes on es mostra cada part del control.



L'estructura del llenguatge de programació es similar a C++, el programa es pot executar en dues parts:

- void setup() es la configuració i es troba sempre al inici del programa, es configuren els pinMode, s'inicialitza la comunicació sèrie, etc i s'executarà una sola vegada després de cada encesa o després d'un Reset amb el polsador de la placa.
- void loop() es l'execució del codi, como lectura d'entrades i activació de sortides.

L'arxiu Configuration.h es on s'allotja la part més important per entendre el funcionament, la Configuració.

Quan hi ha una línia que comença amb dos // o més vol dir que a continuació només són comentaris que ha posat el programador per entendre el seu programa. El programa original està en color verd i els comentaris addicionals per detallar en color vermell.

Si no es tenen coneixements de programació en C++ o C per Arduino, es pot veure una documentació molt ben detallada http://manueldelgadocrespo.blogspot.com/p/se-utiliza-para-la-comunicacion-entre_3.html

```
//////////////////////////////////////////////////////////////////
///// general configuration
//////////////////////////////////////////////////////////////////

// engine to use   Defineix el motor a utilitzar, en aquest cas motor pas a pas
#define ENGINE_TYPE_STEPPERS
```

#define en C es un component útil que permet al programador donar un nom a un valor constant abans de compilar el programa. Les constants definides en Arduino no ocupen cap espai de memòria de programa en el xip. El compilador reemplaça les referències a aquestes constants amb el valor definit en temps de compilació. En Arduino **define** te la mateixa sintaxis que **define** en C:

No hi ha un punto i coma després de la instrucció #define. Si s'inclou un, el compilador generarà errors. Tampoc pot haver un signe (=) a un número valor (5) o (true), per exemple. #

```
// button set to use (analog input, digital input)  Configura el tipus d'entrada que prové de polsadors Analògics o Digitals
#define BUTTONS_ANALOG   Defineix el tipus d'entrada que prové de polsadors Analògics
//#define BUTTONS_DIGITAL  Defineix el tipus de entrada que prové de polsadors Digitals (no està activat)

// milliseconds after a button is considered as pressed  Defineix quants mili-segons, com a mínim, te que estar polsat un botó, para evitar polsos paràsits inesperats en aquesta línia de polsadors.
#define BUTTON_MIN_PRESSED 30     // En aquest cas 30 ms

// milliseconds after a button is considered as long pressed  Defineix quants mili-segons, com a mínim, te que estar polsat un botó, per considerar que es tracte d'una pulsació "llarga", per configurar girs de 60 graus.
#define BUTTON_LONG_PRESSED 1000   // En aquest cas 1000 ms que determina pulsació llarga

// put to false to add movements to the program after its execution
#define PROGRAM_RESET_ALWAYS true   // Fa un Reset dels moviments al final d'executar-se un cicle programat.

// store configuration and program within internal EEPROM
#define USE_PERSISTENT_MEMORY false // Serviria per emmagatzemar en l'Eeprom els moviments, però, no s'utilitza aquesta funcionalitat

// memory capacity for program movements   Defineix la quantitat de moviments que es poden emmagatzemar en la memòria RAM
#define MOVE_LIMIT 100   // En aquest cas es podran emmagatzemar fins a 100 moviments, es pot ampliar el número de moviments programat, ocupant més memòria RAM, que es comparteix amb la que usa el propi programa, però, típicament es suficient. El microcontrolador ATMega328P de la placa Arduino disposa de 2K bytes de RAM i no en sobra molta, però, es podria, si fos necessari, configurar para emmagatzemar més moviments, però, no s'ha provat el límit).

// milliseconds for the "pause" movement     Defineix quants mili-segons tindrà que estar polsat el botó per programar una pausa
#define PAUSE_MOVE_MILLIS 1000    // En aquest cas 1000 ms, que determina una Pausa
```

// milliseconds delay before starting to move Defineix quants mili-segons esperarà el robot per començar a moure's
#define DELAY_BEFORE_GO 500 // En aquest cas començarà a moure's després de 500 ms

// milliseconds to pause after every movement Defineix quants mili-segons tindrà que estar polsat el botó per programar una pausa
#define AFTER_MOVEMENT_PAUSE 0 // En aquest cas començarà a moure's després de 0 ms després de una Pausa

// point of view set when Vacalourabot is started **Escornabot** (abans *Vacalourabot*)
#define POV_INITIAL POV_ESCORNABOT // Punt de vista (Point Of View) de l'execució dels moviments, es a dir des de l'usuari o des del robot, però, no està implementat

// bluetooth serial Defineix el port sèrie que prové del mòdul Bluetooth
#define USE_BLUETOOTH true // Defineix l'ús del mòdul Bluetooth
#define BLUETOOTH_BAUDS 9600 // Defineix la velocitat de transmissió del mòdul Bluetooth

// buzzer Defineix el Brunzidor
#define USE_BUZZER true // Defineix l'ús del mòdul Bluetooth
#define BUZZER_PIN 10 // Defineix el pin usat per el Brunzidor (En aquest cas D10)
#define PROGRAM_FINISHED_RTTL RTTL_FIDO // Buzzer,h // Defineix la musiqueta al **final de l'execució**
#define TONE_FREQ_UP 2637 // Defineix la freqüència de 2637 (que correspon a un Mi octava 7 musical) per senyalitzar que va **endavant**
#define TONE_FREQ_RIGHT 4434 // Defineix la freqüència de 4434 (que correspon a un Do# octava 8 musical) per senyalitzar que **gira a la dreta**
#define TONE_FREQ_DOWN 3520 // Defineix la freqüència de 3520 (que correspon a un La octava 7 musical) per senyalitzar que va **enrere**
#define TONE_FREQ_LEFT 2217 // Defineix la freqüència de 2217 (que correspon a un Do# octava 7) per senyalitzar que **gira a l'esquerra**

Control del Brunzidor

Una sortida PWM del microcontrolador genera freqüències que es poden reproduir en un brunzidor

En aquesta web es poden trobar cada nota musical amb el seu corresponent freqüència i octava corresponent: https://juegosrobotica.es/musica-con-arduino/#

Control del LED

// simple led Defineix el LED
#Define USE_SIMPLE_LED false // Defineix l'ús del LED
#Define SIMPLE_LED_PIN 13 // Defineix el pin 13 pel LED

// keypad leds Defineix els LED dels polsadors
#Define USE_KEYPAD_LEDS true
#Define KEYPAD_LED_PIN_UP A0 // Defineix el pin A0 pel LED per senyalitzar que va **endavant**
#Define KEYPAD_LED_PIN_RIGHT A3 // Defineix el pin A3 pel LED per senyalitzar que **gira a la dreta**
#Define KEYPAD_LED_PIN_DOWN A2 // Defineix el pin A2 pel LED per senyalitzar que va **enrere**
#Define KEYPAD_LED_PIN_LEFT A1 // Defineix el pin A1 pel LED per senyalitzar que **gira a l'esquerra**
#Define KEYPAD_LED_PIN_GO 13 // Defineix el pin 13 pel LED per senyalitzar que es **posa en marxa**

```
//////////////////////////////////////////////////////////////////
///// Steppers engine setup     // Configura els motors pas a pas
//////////////////////////////////////////////////////////////////

#ifdef ENGINE_TYPE_STEPPERS
```

#ifdef permet compilar una secció d'un programa sols si la macro que s'especifica com paràmetre ha estat definida, sense importar el valor que sigui.

```
// stepper pin setup (digital outputs)    Configuració dels pins de control dels motors pas a pas
#Define STEPPERS_MOTOR_RIGHT_IN1 5
#Define STEPPERS_MOTOR_RIGHT_IN2 4
#Define STEPPERS_MOTOR_RIGHT_IN3 3
#Define STEPPERS_MOTOR_RIGHT_IN4 2
#Define STEPPERS_MOTOR_LEFT_IN1 9
#Define STEPPERS_MOTOR_LEFT_IN2 8
#Define STEPPERS_MOTOR_LEFT_IN3 7
#Define STEPPERS_MOTOR_LEFT_IN4 6

// stepper pin setup (digital outputs)    Configuració dels pins de control dels motors pas a pas amb gir invers
#Define STEPPERS_MOTOR_RIGHT_IN1 2
#Define STEPPERS_MOTOR_RIGHT_IN2 3
#Define STEPPERS_MOTOR_RIGHT_IN3 4
#Define STEPPERS_MOTOR_RIGHT_IN4 5
#Define STEPPERS_MOTOR_LEFT_IN1 6
#Define STEPPERS_MOTOR_LEFT_IN2 7
#Define STEPPERS_MOTOR_LEFT_IN3 8
#Define STEPPERS_MOTOR_LEFT_IN4 9

// step calibration     Configuració de la calibratge dels passos
#Define STEPPERS_STEPS_PER_SECOND 1000     // Defineix el número de passos per segon,
```

es pot modificar fins a 2300 passos, però, depèn del motor i també de l'alimentació de la bateria

```
#Define STEPPERS_LINE_STEPS 1738      // Defineix el número de passos per cada pas (10 cm), es pot crear un tapet de joc amb dimensions diferents,
```

1cm serà de 174. A partir d'aquí podem canviar el valor per que avanci el que volem.

```
#Define STEPPERS_TURN_STEPS 1024      // Defineix el número de passos per cada gir (90 graus),
```

per girar a 45 graus es pot modificar a 512 passos, En cas de decimals arrodonir el valor.
Un joc; practicar les hores del rellotge variant els angles de gir.

```
#endif
```

Aquestes directives permeten incloure o descartar part del codi d'un programa si es compleix una determinada condició.

#ifdef permet compilar una secció d'un programa sols si la macro que s'especifica com paràmetre ha estat definida, sense importar quin sigui el seu valor.
Por exemple, si tenim una aplicació i t'agrediria que quan estiguis desenvolupament, sempre inicií amb un usuari determinat.

En aquest cas, aquestes directives et poden ajudar:

```
#Define PRUEBA
```

```
#ifdef PRUEBA
  // CODIGO PARA CARGAR USUARIO POR DEFECTO

#endif
```

El codi de l'interior de la directiva sols serà compilat si PRUEBA està definida. Així quan vagis a publicar la teva aplicació, sols tindries que eliminar la definició de PRUEBA i el programa seguirà el seu flux normal.

Control dels Polsadors

Com s'ha mostrat al principi els polsadors estan enllaçats amb resistències a un sols pin, en l'arxiu Configuration.h

Per el que aquesta part de configuració Digital no està activada

```
//////////////////////////////////////////////////////////////////////
///// Button set digital
//////////////////////////////////////////////////////////////////////

#ifdef BUTTONS_DIGITAL

// keypad pin setup (digital or analog inputs) (use 255 if key doesn't exist)
#Define BS_DIGITAL_UP A0
#Define BS_DIGITAL_RIGHT A1
#Define BS_DIGITAL_DOWN A2
#Define BS_DIGITAL_LEFT A3
#Define BS_DIGITAL_GO A4
#Define BS_DIGITAL_RESET 255

#endif // BUTTONS_DIGITAL
```

Aquesta part si que funciona

```
//////////////////////////////////////////////////////////////////////
///// Button set analog
//////////////////////////////////////////////////////////////////////

#ifdef BUTTONS_ANALOG

#Define BS_ANALOG_WIRES 2   // WIRE 2 ?
//#Define BS_ANALOG_WIRES 3    // no s'usa

// keypad pin setup (analog input)
#Define BS_ANALOG_PIN A7    // Configura el pin A7 com entrada dels Polsadors

// input values for each key pressed (0 if key doesn't exist)
#Define BS_ANALOG_VALUE_UP 512    // Valors para Polsador Endavant
#Define BS_ANALOG_VALUE_RIGHT 860  // Valors para Polsador Dreta
#Define BS_ANALOG_VALUE_DOWN 769  // Valors para Polsador Enrere
#Define BS_ANALOG_VALUE_LEFT 683  // Valors para Polsador Esquerra
```

```
#Define BS_ANALOG_VALUE_GO 810  // Valors per Polsador  GO
#Define BS_ANALOG_VALUE_RESET 0  // Valors per RESET ?

#endif // BUTTONS_ANALOG


//////////////////////////////////////////////////////////////////
///// Button set Bluetooth   //Configuració del Bluetooth
//////////////////////////////////////////////////////////////////

#ifdef USE_BLUETOOTH

// Arduino serial port (default is Serial, use Serial1 with Arduino Micro)
//#Defineix BLUETOOTH_SERIAL    Serial    si trèiem les barres  // ja deixa preparat per rebre dades a través del mòdul Bluetooth
//#Define BLUETOOTH_SERIAL    Serial1
//#Define BLUETOOTH_SERIAL    Serial2
//#Define BLUETOOTH_SERIAL    Serial3

#endif // USE_BLUETOOTH
```

Modo de joc a 90 o 60 graus

El firmware està preparat per treballar a 90 graus i a 60 graus amb pulsació llarga. La selecció del tipus de joc ve definida en l'archivo Enums.h

```
// game modes
enum
{
   GAME_MODE_GRID_90 = 0, // squared grid (classical mode)
   GAME_MODE_GRID_60 = 1, // triangled grid
};
typedef uint8_t GAME_MODE;
```

Programació dels moviments de "Salutació"

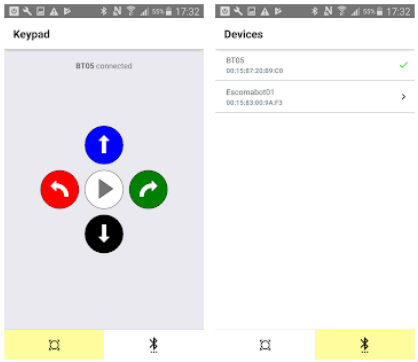
Els moviments de "salutació" inicial al alimentar l'Escornabot, estan programats en la memòria Eeprom del microcontrolador, on queden emmagatzemades encara que es tregui de nou l'alimentació i en el firmware està en l'arxiu MoveList.h

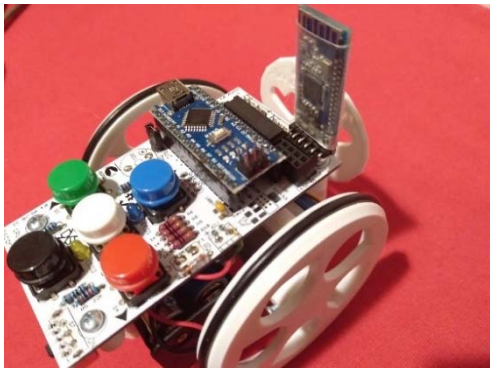
```
const static MOVE PROGRAM_ESCORNA_GREETING[] = {
   MOVE_FORWARD,    // Mou cap endavant
   MOVE_LEFT,       // Mou cap a l'esquerra
   MOVE_RIGHT,      // Mou cap a la dreta
   MOVE_RIGHT,      // Mou cap a la dreta
   MOVE_LEFT,       // Mou cap a l'esquerra
   MOVE_BACKWARD,   // Mou cap enrere
   MOVE_NONE
};
```

Si es volen programar altres moviments, sols hi ha que modificar, treure o afegir las línies que es mostren en l'arxiu MoveList.h . El número de moviments como es pot veure són 6, però, la memòria Eeprom permet una quantitat elevada de moviments, te 1 KBytes

Escornabot segueix creixent

A més, Escornabot està preparat per afegir un mòdul de Bluetooth tipus HM-10 amb la App disponible per Android i IOs.





També està preparat per un mòdul Wifi, encara que no es recomanable típicament per l'alt consumo amb respecte al mòdul Bluetooh, però, es possible.

El control remot no es recomanable per els usuaris de menor edat, ja que es millor treballar sobra d'una taula o en el terra. En canvi, l'ús de control remot permet realitzar un altre tipus de pràctiques com el disseny d'un control des d'un telèfon amb AppInventor.

EngineSteppers.cpp

Es on fa girar en un sentit o un altre els motors

```
void EngineSteppers::init()
{
   pinMode(_config->motor_left_in1, OUTPUT);
   pinMode(_config->motor_left_in2, OUTPUT);
   pinMode(_config->motor_left_in3, OUTPUT);
   pinMode(_config->motor_left_in4, OUTPUT);
   pinMode(_config->motor_right_in1, OUTPUT);
   pinMode(_config->motor_right_in2, OUTPUT);
```

```
  pinMode(_config->motor_right_in3, OUTPUT);
  pinMode(_config->motor_right_in4, OUTPUT);

  // set coils in row // just al alimentar fa aquest petit moviment per estabilitzar la posició de las bobines dels motors
  _movement_steps_r = 8;
  _movement_steps_l = 8;

  EVENTS->add(this);
}
```