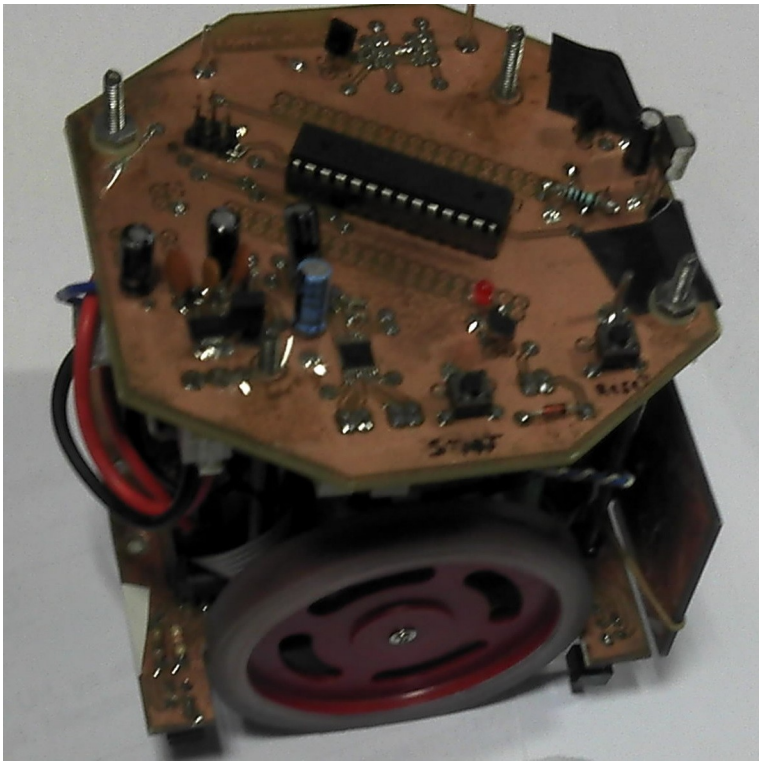
Competencia de robótica

JOHNNY 5

~2015~

Categoría: **Minisumo**

Nombre robot: **Skil**



Institución: **Club de Robótica - FIUBA**

Participantes:

- **Ariel Burman** **arielburman@gmail.com**
- **Sebastián García Marra**
- **Joaquín de Andrés**

Introducción

Skil es un robot minisumo desarrollado en el Club de Robótica de FIUBA. Su nombre surgió en base a que todas las partes antes de ser construido se guardaban en una caja de un taladro Skil.

Toda la información del proyecto puede accederse en
`http://labi.fi.uba.ar/chiliproject/projects/skil`

Diseño y desarrollo

El robot posee un led emisor y un receptor de luz infrarroja que se utiliza para detectar al oponente. Cuando la luz emitida rebota en una superficie y vuelve hasta el robot, es detectada con el receptor. El receptor detecta señales moduladas en 38 kHz, por lo cual para excitar el led se genera un tren de pulsos de frecuencia 38 kHz.

El robot posee 4 sensores infrarrojos cercanos al piso. Cada sensor detecta si la superficie es negra o blanca. Cuando alguno de los 4 sensores se activa, el robot ignora cualquier otro sensor y se mueve en la dirección opuesta a la línea blanca detectada.

Cuando el robot detecta un oponente avanza a máxima potencia con la intención de sacarlo del tatami. Si el robot no detecta un oponente, ni está cerca del borde del tatami, realiza una de 3 acciones: avanza, gira a la derecha o gira a la izquierda. La selección de cuál de las 3 acciones realiza, es aleatoria.

Mecánica

Figura 1: Motores SolarBotics

Figura 2: Baterías de Litio - Ion

El tamaño del robot se ajusta a las dimensiones establecidas en los reglamentos de 10 cm x 10 cm. La estructura mecánica se compone de 2 PCB unidos por 4 tornillos pasantes que los unen entre sí. Entre los dos PCB se encuentran los motores (figura 1) Solarbotics[1] con reducción 120:1 y dos baterías[2] de Litio-Ion de 3,7 V cada una, dando lugar utilizando una alimentación

[1] https://www.pololu.com/product/1121
[2] https://www.pololu.com/product/1003

Figura 3: Ruedas Solarbotic 70 mm

Figura 4: Goma de silicona de mayor adherencia

de 7,4 V (figura 2). Utiliza ruedas de 70 mm de Solarbotics[3] (figura 3), con llantas de silicona de mayor adherencia [4] (figura 4).

Se agregaron pilas en desuso para aumentar el peso del robot hasta alcanzar 500 g, que es el límite de la categoría.

Electrónica

En la figura 5 se encuentra el circuito esquemático del PCB inferior, que contiene los 4 sensores infrarrojos CNY70 que permiten detectar el borde del tatami.

En la figura 6 se encuentra el circuito esquemático del PCB de la placa principal. El circuito incluye el microcontrolador ATmega88PA[5], el controlador de los motores, puente-H DRV8833[6] y el emisor y receptor infrarrojo que permite la detección del oponente. Este circuito contemplaba la posibilidad de incluir un kit de desarrollo mBed[7], aunque nunca se llevó a cabo.

[3]https://www.pololu.com/product/184
[4]https://www.pololu.com/product/694
[5]http://www.atmel.com/devices/atmega88pa.aspx
[6]http://www.ti.com/product/drv8833
[7]www.mbed.org

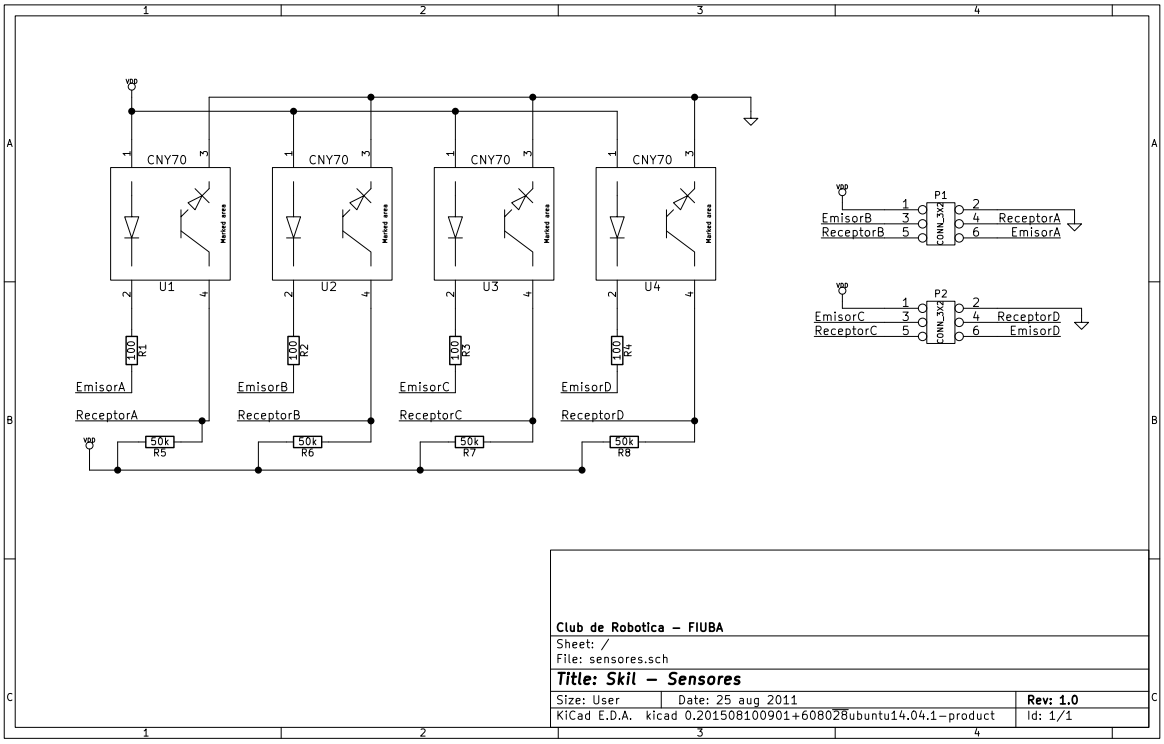


Figura 5: Circuito esqumático del PCB inferior

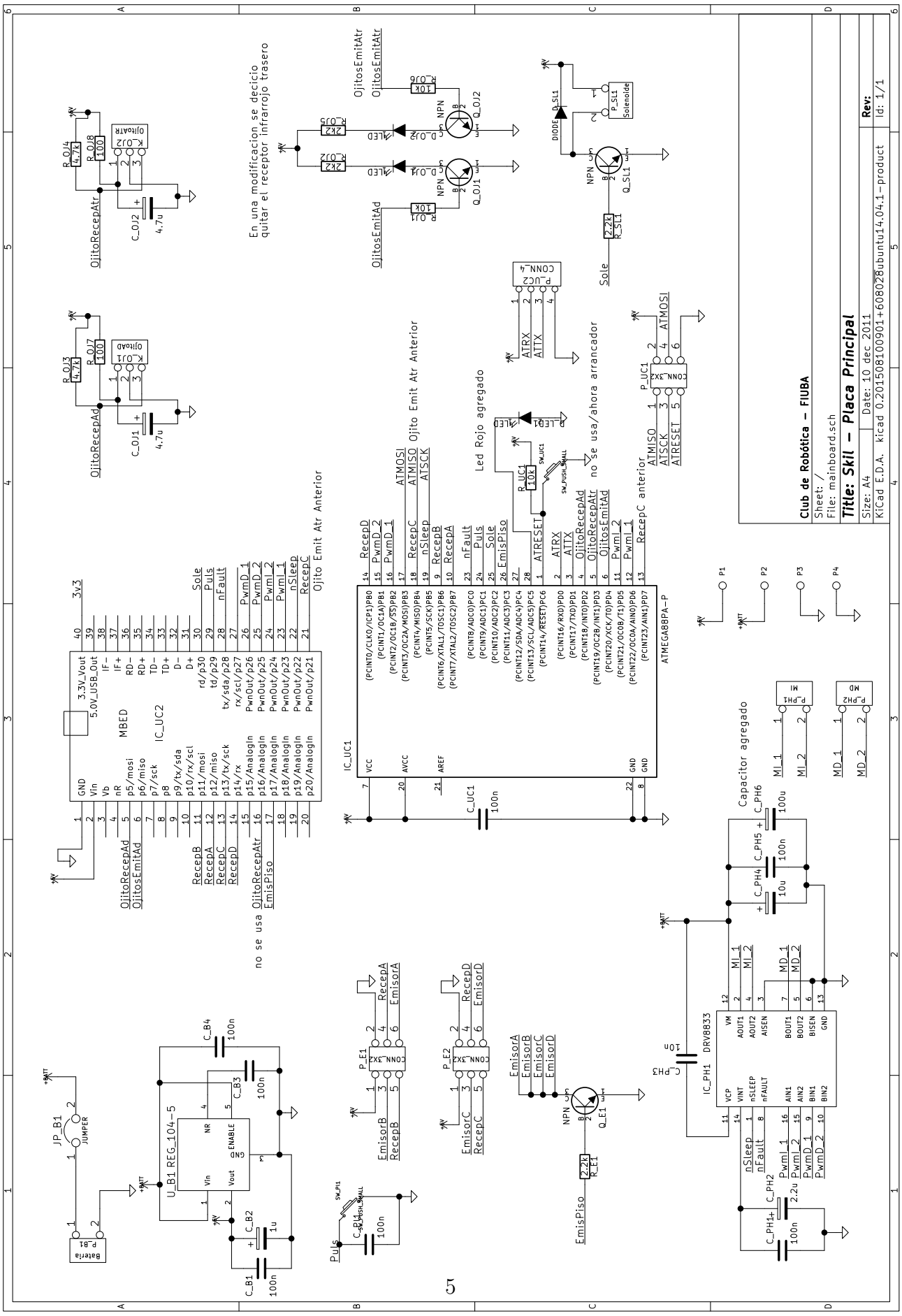


Figura 6: Circuito esquemático del PCB superior

Código Fuente

Se incluye el código fuente del programa principal. Todo el código fuente se puede encontrar en la página del proyecto

`http://labi.fi.uba.ar/chiliproject/projects/skil`

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "motores.h"
#include "sensores.h"
#include "skil.h"

#define DELAY_ESTADO     2 // en milisegundos
#define DELAY_ESCAPE     500 // en milisegundos
#define DELAY_INICIO     200 // en milisegundos 5100
#define ALEATOREO_DELAY    500

// la macro
#define proximoEstadoAleatorio() (TCNT2 & 0x03) //Devuelve un numero aleatorio entre 0 y 3
#define ALEATOREO_MAX_CUENTA    5000

volatile estados_t estado;

// utilizado para definir la cantidad de repeticion de la misma accion aleatorio
// (ver accionTracking )
volatile uint8_t cantVeces;

int main (void) {
        setup();

//       if (IsPulsadorSet() == false) {
//
//     desenergizarSolenoide();
//     EncenderMotores();
//     cli();
//     while(1) movimientoPrueba();
//   }
  _delay_ms(1000);
  while(estado==DETENIDO);
  Led1On();
  while ((PIN_REMOTO & (1<<REMOTO_NUMBER)) == 0);
  Led1Off();

  //tiempo de espera para bajar pollera
  //_delay_ms(DELAY_INICIO); //sacamos

  // la pollera no la estamos usando, pero mejor desconectarla
  // el solenoide no puede que
  desenergizarSolenoide();

  encenderEmisorSuperior();
  encenderEmisorInferior();
  _delay_ms(5);
```

```
    while(1){

// Verifica el puente H
if (IsnFaultSet()==0) {
  titilarLed(20);
  ConfigurarMotores();
  EncenderMotores();
  MoverAdelante();
}
if ((PIN_REMOTO & (1<<REMOTO_NUMBER)) == 0) ApagarMotores();
sensoresInf = PINB & MASK_INT_PIN_ALL;


switch (sensoresInf){
  case OK:
    switch(estado){
      case FIGHT_ADELANTE:
        Led1On();
              MoverAdelante();
        break;
      case TRACKING:
        Led1Off();
        accionTracking();
        break;
      case ESCAPE:
        Led1Off();
        break;
      case DETENIDO:
      default:
        ApagarMotores();
        titilarLed(12);
        break;
    }
    break;
  case ATRAS:
    estado = ESCAPE;
    arrancarTimerEstados(0x6000);
    MoverAdelante();
    break;
  case ADELANTE:
    estado = ESCAPE;
    arrancarTimerEstados(0x9000);
    MoverAtras();
    break;
  case ADELANTE_DER:
    estado = ESCAPE;
    arrancarTimerEstados(0x6000);
    GirarIzquierdaAtras();
    break;
  case ADELANTE_IZQ:
    estado = ESCAPE;
    arrancarTimerEstados(0x6000);
    GirarDerechaAtras();
    break;
```

```
      case ATRAS_DER:
        estado = ESCAPE;
        arrancarTimerEstados(0x1000);
        GirarIzquierdaAdelante();
        break;
      case ATRAS_IZQ:
        estado = ESCAPE;
        arrancarTimerEstados(0x1000);
        GirarDerechaAdelante();
        break;
      case DERECHA:
        estado = ESCAPE;
        arrancarTimerEstados(0x1000);
        GirarIzquierdaAdelante();
        //RotarIzquierda();
        break;
      case IZQUIERDA:
        estado = ESCAPE;
        arrancarTimerEstados(0x1000);
        //RotarDerecha();
        GirarDerechaAdelante();
        break;
      default:
        //ApagarMotores();
        //titilarLed(10);
        break;
    }
  }
}


void setup (void) {
        ConfigurarMotores();
        configurarPinSensoresSup();
        configurarPinSensoresInf();
        configurarTimerSensoresSup();
        configurarTimerEstados();
        configurarPulsador();
        configurarRemoto();
  configurarSolenoide();
//  energizarSolenoide();
  Led1Init();
  Led1Off();

        estado = DETENIDO;
  cantVeces = 0;
        sei();
}

void configurarPulsador(void){
        PulsadorInit();
        // Configuro el pin change
        PCICR |= (1<<PCIE1);
        PCMSK1 = (1<<PCINT9);
}
```

```
void configurarRemoto(void){
  ClearBit(DDR_REMOTO, REMOTO_NUMBER);
  SetBit(PORT_REMOTO, REMOTO_NUMBER);
}

void accionTracking(void){
        uint8_t temp;
  if (cantVeces < ALEATOREO_MAX_CUENTA) {
    cantVeces++;
  }
  else{
    cantVeces=0;
    temp = proximoEstadoAleatorio();
    switch(temp) {
      case 0:
        RotarIzquierda();
        break;
      case 1:
        GirarDerechaAdelante();
        break;
      case 2:
        GirarDerechaAtras();
        break;
      case 3:
        RotarDerecha();
        break;
    }
  }
}

// Interrupcion de pulsador (vetor_4)
ISR(PCINT1_vect) {
        // Delay para debounce
        // Dado que no tenemos necesidad de hacer nada mientras esperamos por el
        // debounce lo dejamos asi. Sino, deberiamos utilizar algun timer
        _delay_ms(250);

        if (IsPulsadorSet() == true) {
        // significa que esta en 1 y hubo flanco ascendente genuino
        // se podria reemplazar la variable por poner apagar todo, poner
        // el micro a dormir esperando solo esta interrupcion y luego
        // despertalo. Aca se lo despertaria
    EncenderMotores();

    //estado = TRACKING;
    estado = FIGHT_ADELANTE;
        }

  // Este flag se clerea a mano porque el clear por hardware se realiza en el
  // momento que se atiende la interrupcion y no cuando se sale de ella.
  // Esto hace que mientras se esta dentro de la interrupcion, puedan generarse
  // nueos flancos (ruido) que no queremos atender
  SetBit(EIFR, INTF0);
}
```

```
/* funciones de prueba */
void movimientoPrueba(void){
        MoverAdelante();
        _delay_ms(400);
        GirarIzquierdaAtras();
        _delay_ms(400);
        GirarDerechaAtras();
        _delay_ms(400);
        MoverAtras();
        _delay_ms(100);
}

void titilarLed(uint8_t numero){
  uint8_t i;

  Led1Off();
  for (i=0;i<(2*numero);i++){
    Led1Toggle();
    _delay_ms(200);
  }
    _delay_ms(300);
}
```