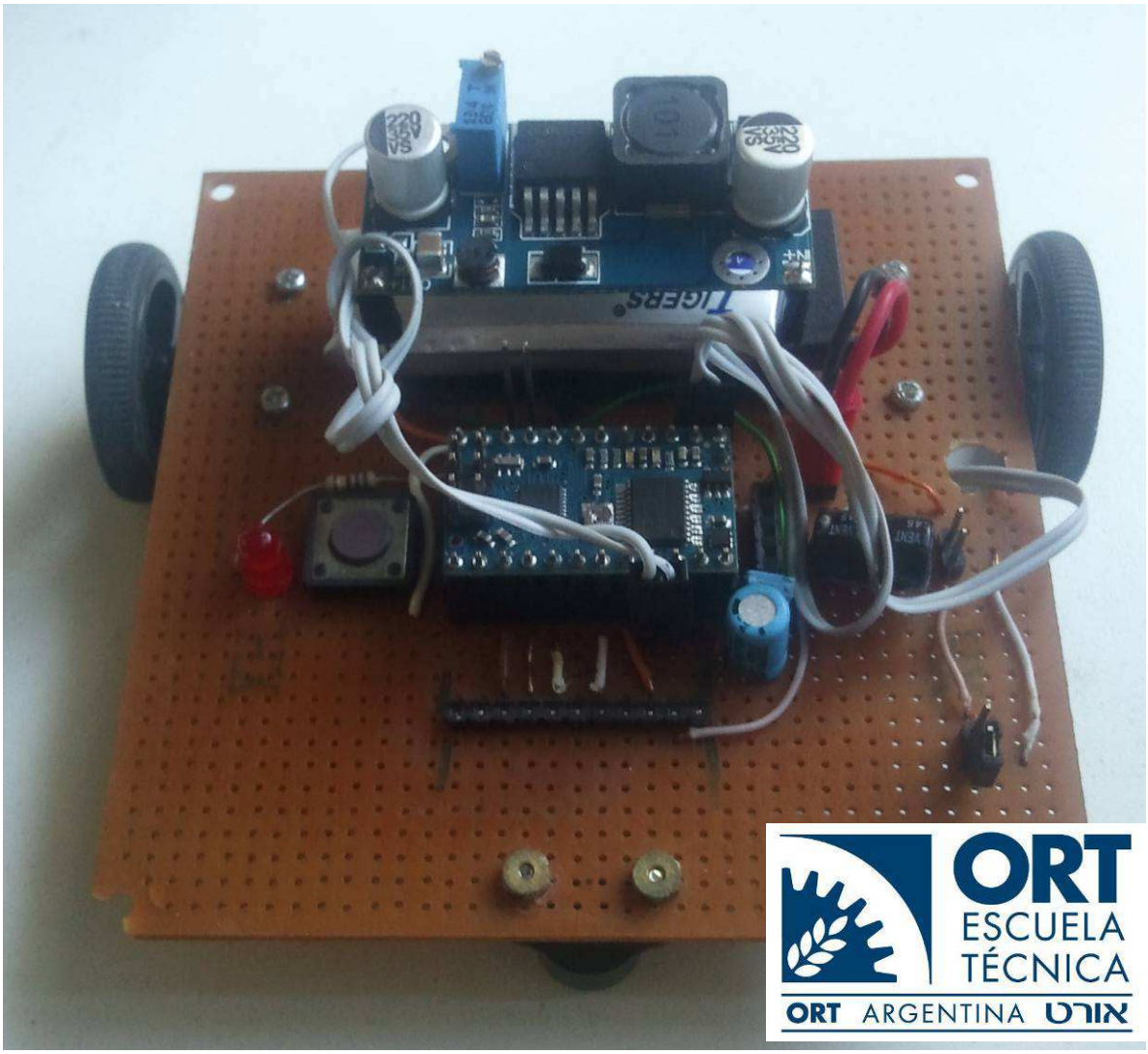
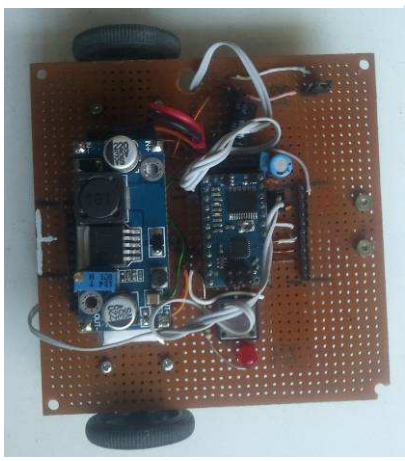
Competencia de Robótica **“R2D2”**. Facultad de Ingeniería
Universidad de Buenos Aires – 6 de Septiembre de 2014

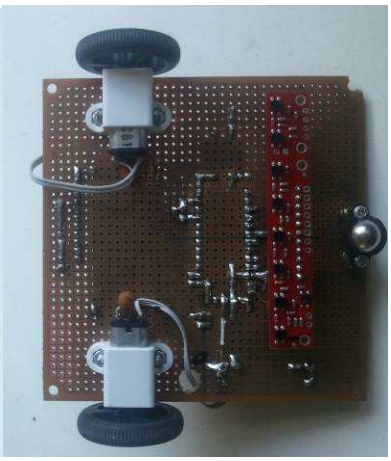
Pingüino

Escuela Técnica ORT, sede Almagro



Mecánica


Vista superior


Vista Inferior

- Dimensiones: 120 x 100mm. Altura 60mm
- Ruedas de 35mm de diámetro
- Estructura: plaqueta universal para prototipos electrónicos (100 x 100 mm)
- Tracción diferencial (dos motores en la parte posterior)
- Punto de apoyo, tipo ball-caster
- Peso: 105 g

Electrónica

- Controlador Baby Orangutan (circuito esquemático en anexo)
- Elevador DC-DC (circuito esquemático en anexo)
- Sensores reflectivos IR QTR-RC (circuito esquemático en anexo)
- Modulo UART inalámbrico (WIXEL).

Funcionamiento básico

Básicamente el funcionamiento tiene 4 etapas:

Calibración de los sensores:

Se utilizan 5 de los 8 sensores presenten en el modulo, cada sensor responde de manera parecida pero no idéntica al color detectado (emiten luz infrarroja y miden la intensidad de la luz recibida). Al presionar el pulsador, el robot barre 180º con sus motores, y durante ese movimiento captura muchas muestras de cada sensor, obteniendo al final un valor máximo y otro mínimo para cada uno individuamente. Estos valores son almacenados en RAM y luego utilizados.

Luego, se presiona una vez más el pulsador y el programa entra en un loop continuo ejecutando 2 funciones principales:

Adquisición, ajuste y procesamiento de la posición:

Se leen los sensores, utilizando los máximos y mínimos obtenidos en el proceso de calibración, se ajusta la lectura de cada sensor, con una ecuación lineal a un valor entre 0 y 1000 (0 totalmente negro – 1000 totalmente blanco). Con esos 5 valores entre 0 y 1000, se realiza una cuenta, ponderando la posición de los sensores y su valor, para obtener finalmente un número entero entre 0 y 4000 que indica el lugar donde se encuentra el robot respecto de la línea, siendo 0, 1000, 2000, 3000 y 4000 los valores para los cuales el sensor 0, 1, 2, 3, 4 está justo arriba de la línea blanca respectivamente.

Aplicación de ecuación de corrección:

Al ser el sensor 2 el “centro” del robot, nuestro objetivo es mantener el valor de la lectura en 2000. Para esto se resta el valor 2000 al valor calculado (Línea), quedando un valor entre -2000 y 2000, siendo los extremos derecho e izquierdo, y el centro “0”.
Este valor (entre -2000 y 2000) indica cuan lejos estamos del valor ideal (0), por lo que lo multiplicamos por un factor que indica que debemos hacer con los motores en función de cuan alejados estamos del centro. A esto se lo llama constante proporcional. Además, se calcula la diferencia entre los dos últimos valores de la linea, y se obtiene un termino derivativo, que nos permite tener una idea de la variación entre una lectura y la siguiente. Este factor, multiplicado por la constante derivativa, se suma también a la corrección.

Modificación de velocidad de los motores:

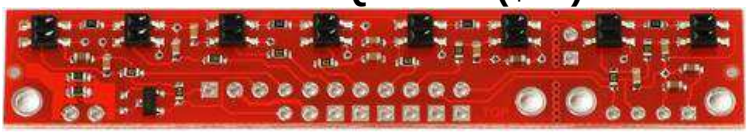
El valor obtenido se considera la diferencia de velocidad entre el motor derecho e izquierdo. Un motor siempre irá al máximo y el otro más lento (según lo calculado por la ecuación)

NOTA: se cuenta con la posibilidad de enviar valores por la UART para debug de forma inalámbrica (modulo WIXEL)

Componentes y precios

Costo total: $552

Tira de 8 sensores QTR-RC ($80)

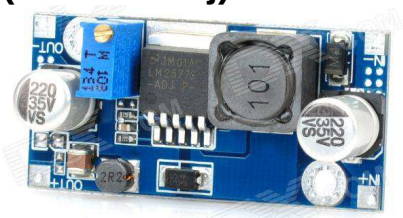


http://www.pololu.com/catalog/product/961

Controlador Baby Organgutan ($106)
(Atmel 328p + driver de motores TB6612FNG)

http://www.pololu.com/catalog/product/1220

Elevador de tensión DC-DC ($40)
(LM2577-adj)



http://dx.com/p/dc-3-30v-to-dc-4-35v-adjustable-boost-converter-charger-module-148492

Micro Motores 30:1 ($170)

http://www.pololu.com/catalog/product/1098

Soporte para motores ($26)

http://www.pololu.com/catalog/product/1089

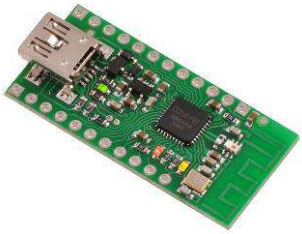
Ruedas ($42)

http://www.pololu.com/catalog/product/1420

Punto de apoyo – rueda loca ($16)

http://www.pololu.com/catalog/product/951

Modulo Wixel ($106)



http://www.pololu.com/catalog/product/1337

Bateria LiPo 3,7v 600mAh ($16)

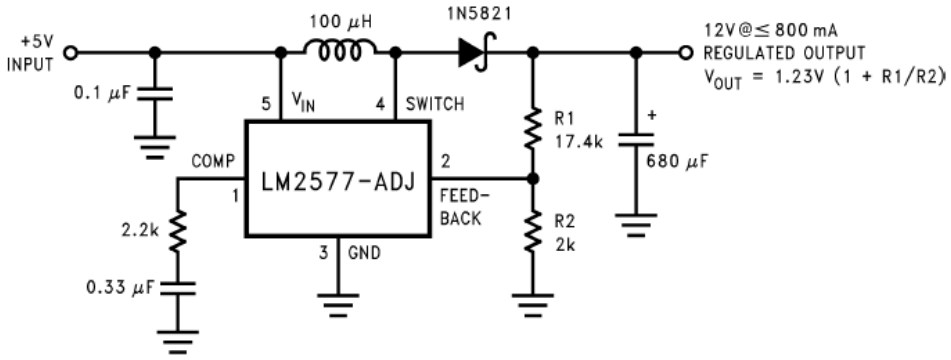
http://dx.com/p/tiger-5001s-1s-3-7v-15c-600mah-li-ion-polymer-battery-pack-for-r-c-aircraft-more-3-pcs-180594

Plaqueta experimental (10x10cm) ($30)

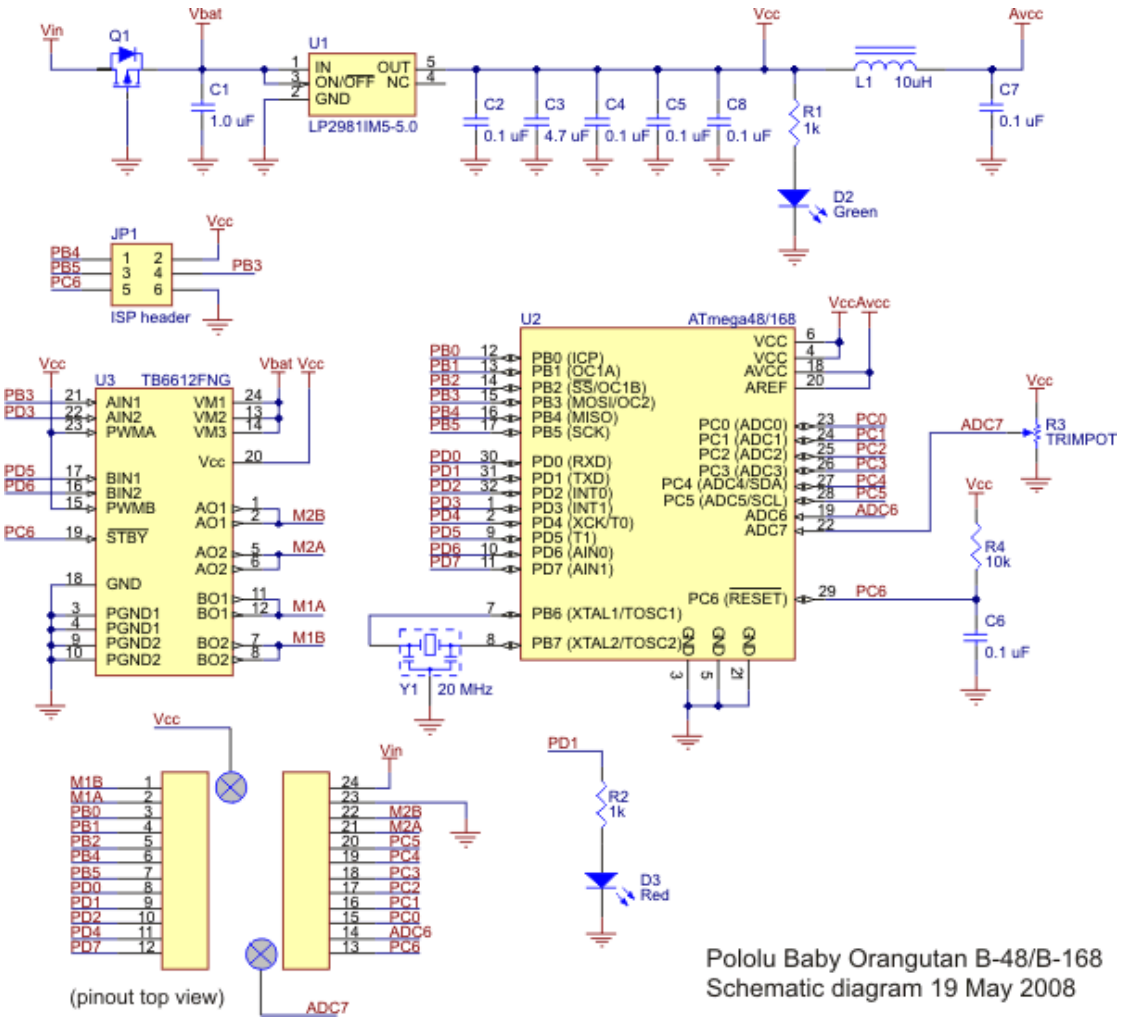
http://articulo.mercadolibre.com.ar/MLA-460698723-plaquetas-experimentales-pertinax-10x10cm-high-tec-electroni-_JM

Circuitos Esquemáticos

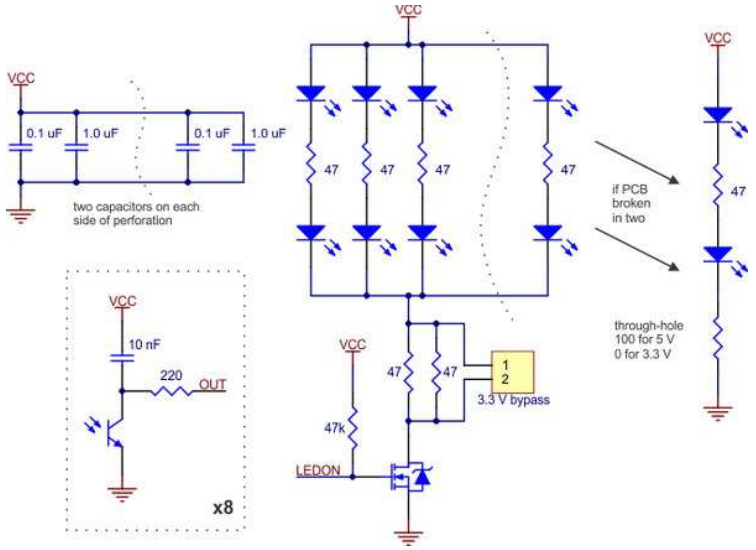
Elevador de tensión DC-DC



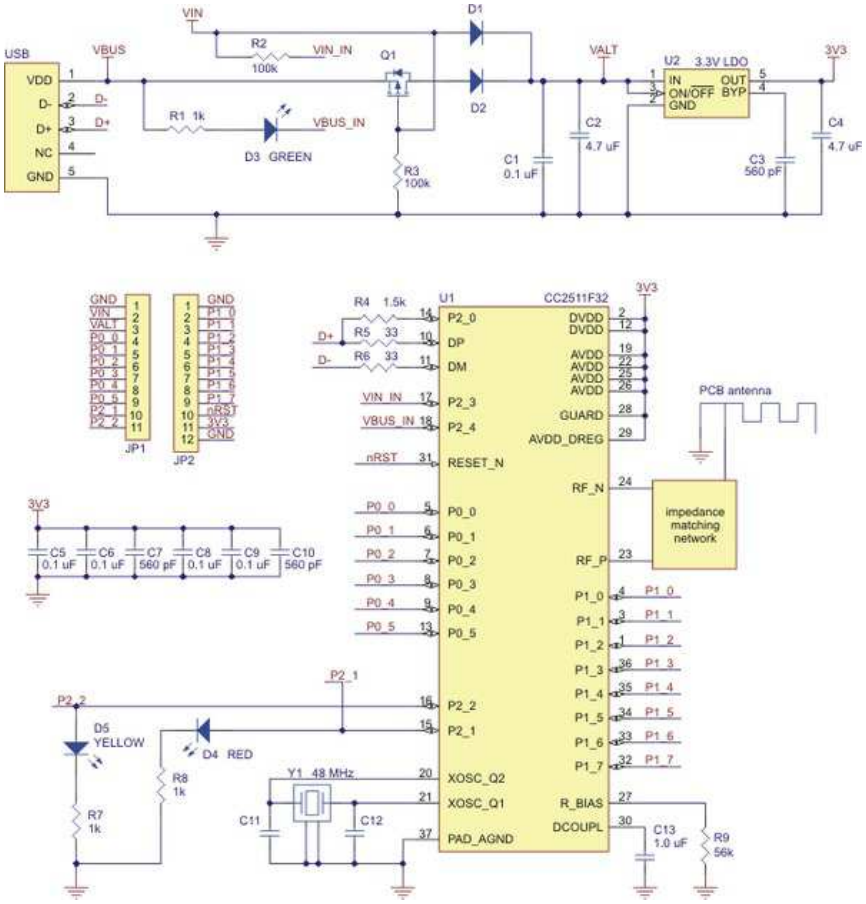
Controlador Baby Organgutan



Tira de 8 sensores QTR-RC



WIXEL



Código fuente

```
#include <pololu/orangutan.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define LED IO_D4

#define STATE_RECTA 1
#define STATE_CURVA 2
#define STATE_START 3

#define SPEED_VALUE_MAX_RECTA 170
#define SPEED_VALUE_MAX_CURVA 100
#define SPEED_VALUE_BASE 130

void hola(void);
void leer(void);
void calibrar(void);
void wait_for_sending_to_finish();

int main()
{
	unsigned int last_proportional=0;
	unsigned int contador=0;
	long integral=0;

	//unsigned char qtr_rc_pins_3[]={IO_C3,IO_C2,IO_C1};
	unsigned char qtr_rc_pins_5[]={IO_C4,IO_C3,IO_C2,IO_C1,IO_C0};

		char send_buffer[64];//,aux[64];

	unsigned int pos;
	int proportional;
	int derivative;
	unsigned int sensores[5];

	float p = 0.1,d = 0,i = 0;
	int speed_state;
	int max;
	int contador_recta = 0;
	int contador_curva = 0;
	int contador_estable = 0;

	serial_set_baud_rate(57600);

	set_digital_input(IO_D7, PULL_UP_ENABLED);


	hola();


	delay_ms(200);

	sprintf(send_buffer, "Hola Mundo\r");
	wait_for_sending_to_finish();
	serial_send(send_buffer, strlen(send_buffer));
```

```
qtr_rc_init(qtr_rc_pins_5,5,2000,IO_C5);
set_digital_output(IO_C5, LOW);

while(is_digital_input_high(IO_D7)){}
hola();
calibrar();

contador=0;
while(is_digital_input_high(IO_D7))

        {
                pos=qtr_read_line_white(sensores,QTR_EMITTERS_ON);
                proportional = ((int)pos) - 2000;  //OJO! si uso 3 sensores el valor medio es 1000
                proportional=proportional/10;
                proportional=proportional*10;
                sprintf(send_buffer, "%i\r",proportional);
                wait_for_sending_to_finish();
                serial_send(send_buffer, strlen(send_buffer));
                delay_ms(20);
                set_digital_output(LED, TOGGLE);
        }

sprintf(send_buffer, "Arranca!\r");
wait_for_sending_to_finish();
serial_send(send_buffer, strlen(send_buffer));
hola();
while(!is_digital_input_high(IO_D7));
delay(20);

set_digital_output(LED, LOW);
set_motors(100,100);
delay(100);// primer impulso anti cuelgue

while(1)
{
        //------------ LEO LINEA ----------
        pos=qtr_read_line_white(sensores,QTR_EMITTERS_ON);

proportional = ( (int)pos ) - 2000;  //OJO! si uso 3 sensores el valor medio es 1000

        proportional=proportional/10;
        proportional=proportional*10;

        derivative = proportional - last_proportional;
        last_proportional = proportional;
        integral = integral + proportional;


        //========== SETEO VELOCIDAD ========
        p = 5;
        d = 0.005;
        max=255;


        //===============
int power_difference = (float)proportional * p  + (float)derivative*d;
        //=================
```

```
                if(power_difference > max)
                power_difference = max;

                if(power_difference < -max)
                power_difference = -max;

                if(power_difference < 0)
                set_motors(max+power_difference, max);
                else
                set_motors(max, max-power_difference);
                //================================

                // ===== mando debug
                //sprintf(send_buffer, "%ld\r",integral);
                //sprintf(send_buffer, "%d\r",proportional);
                //sprintf(send_buffer, "%d\r",power_difference);
           //wait_for_sending_to_finish();
                //serial_send(send_buffer, strlen(send_buffer));
                //=============
                //delay(10);
        }
}

void hola(void)
{
        unsigned char i;
        for (i=0;i<5;i++){

                set_digital_output(LED, HIGH);     // Turn on the red LED.
                delay_ms(100);  // Wait for 200 ms.

                set_digital_output(LED, LOW);     // Turn off the red LED.
                delay_ms(100);  // Wait for 200 ms.
        }
}

void calibrar(void)
{
        int counter;
        for(counter=0;counter<80;counter++)
        {
                if(counter < 20 || counter >= 60)
                set_motors(50,-50);
                else
                set_motors(-50,50);
                qtr_calibrate(QTR_EMITTERS_ON);
                delay_ms(8);
        }
        set_motors(0,0);
}



void wait_for_sending_to_finish()
{
        while(!serial_send_buffer_empty());
}
```