EnduranceRobots.com
gf@endurancerobots.com

FULL GUIDANCE AND TUTORIAL: HOW TO MAKE A DIY SELFIEBOT

© ENDURANCE 2016

Content

Electronics assembling process

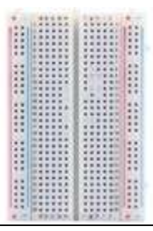
Brief instruction for a DIY SelfieBot (SelfieBot Base) assembly process.

Breadboard: arrange and check the following items:

- ✓ A contact board (830 contacts, type MB-102) - 1 pc.
- ✓ Connecting wires (10cm, male-to-male) - 21 pcs.
- ✓ A microcontroller Arduino Nano - 1 pc.
- ✓ A Bluetooth module HC-06 - 1 pc.
- ✓ DC / DC converters (XL-4005) - 2 pcs.
- ✓ A power connector (such as dc-005) - 1 pc.
- ✓ A fuse (2A) - 1 pc.
- ✓ Connectors for servos (PLS 3pin, 2,54mm pitch, 15mm length) - 2 pcs.
- ✓ Connectors for connecting jumpers (PLS 2pin, 2,54mm pitch, 15mm length) - 2 pcs.
- ✓ Resistors on the pin connector (100 ohms, PLS 2pin, 2,54mm pitch) - 2 pcs.
- ✓ Jumpers (2pin, 2,54mm pitch) - 2 pcs.

List of parts

Model name	**Picture**	**Amount**	**Basic kit** electronics	**Pre-assembled kit**
USB Nano V3.0 ATmega328 16M 5V Micro-controller CH340G board For Arduino		1	X	X
Wireless Serial 4 Pin Bluetooth RF Transceiver Module HC-06 Slave for Arduino DH		1	X	X
A1) Main DC-DC Converter. LM2596 Step Down Module DC-DC Buck Converter Power Supply Output 1.3V-35V		2	X	X
A2) Alternative DC-DC Converter. Mini Adjustable DC-DC Converter Step down Power Supply Module 3A LM2596		—	X	X

MG996R Torque Digital All Metal Gear Servo for Helicopter Car Boat Model BE		2	X	X
Fuse pin, H630PT, 1 A, 250 V		1	X	X
Resistor, 0.5 Wt, 100 OHM		3	X	X
CONN JUMPER SHORTING, 2.54x8.5mm, 2 contacts		1	X	X
Power jack on the board, 2.1mm, 16V/1.5A		1	X	X
AC Adapter Power Supply DC12V 1-2 A, plug 5.5x2.1		1	X	X
A1) Prototyping Experimental 400 Contact Holes Solderless Breadboard Test Plate		2	X	X

A2) Alternative Prototyping PCB Universal Board 70x90 Double-Sided Glass Fiber Prototyping PCB Universal Board		1	X	X
10pcs 1x 40Pin 40p Male IC Single Row Flat Header Socket 2.54mm PLS-40 Panel		1	X	X
Hot Flexible 16AWG ~30AWG Stranded UL1007 Wire Cable Cord Hook-up DIY Electrical		1	X	X

Mold

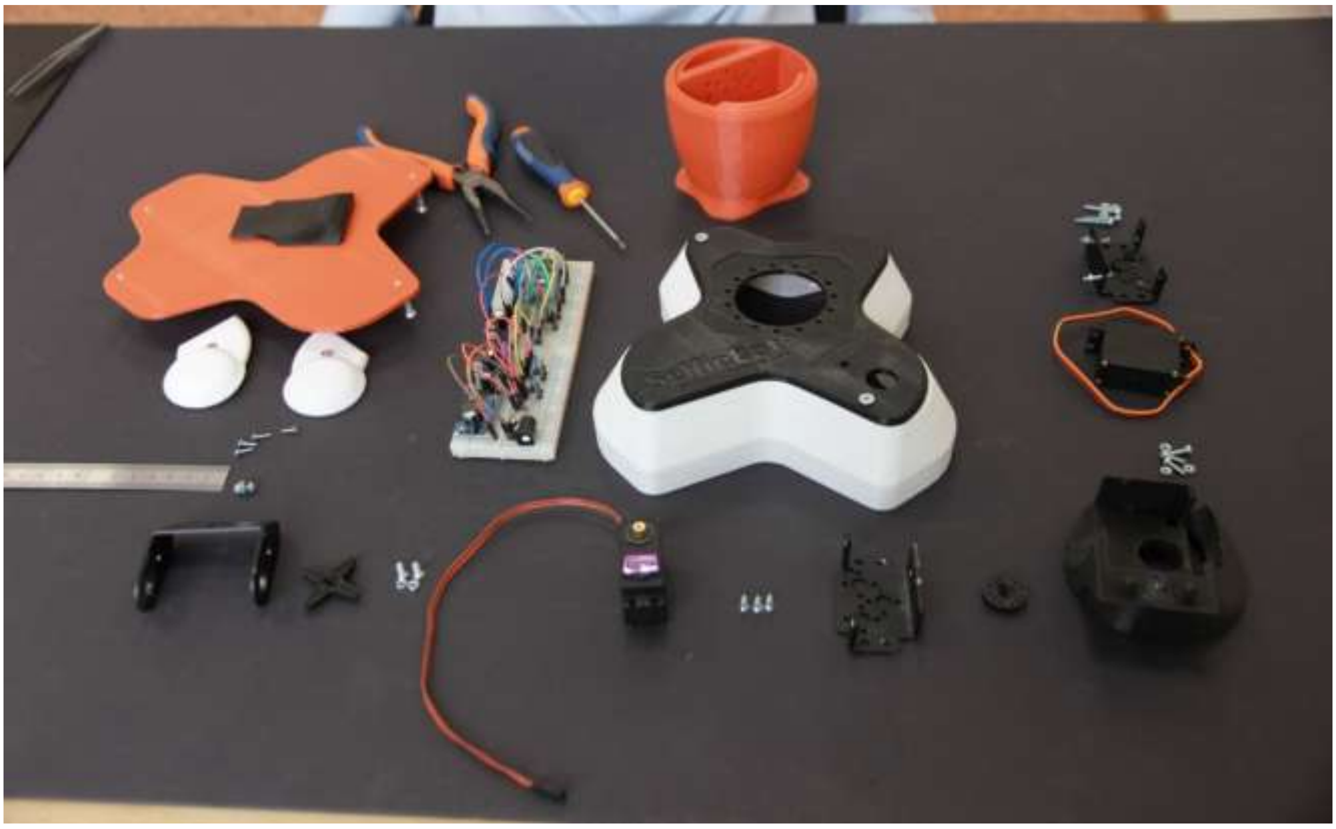
Model name	**Picture**	**Amount**	**Basic kit** electronics	**Pre-assembled kit**
A1) Plastic mobile device cradle/holder (BY DEFAULT)		1		X
A2) Magnet mobile device cradle/holder (ALTERNATIVE)		1		X

3D Printer Filament USA 1.75mm ABS PLA 1KG / 2.2LB		1		
screwnuts 3 mm diameter, 26 pcs.		—		X
phillips countersunk head screws, 3 mm diameter: 10 mm — 8 pcs. 16 mm — 8 pcs. phillips round head screws, 3 mm diameter: 8 mm — 2 pcs. 16 mm — 6 pcs. 20 mm — 10 pcs. phillips countersunk head screws, 4 mm diameter: 25 mm — 6 pcs.		—		X

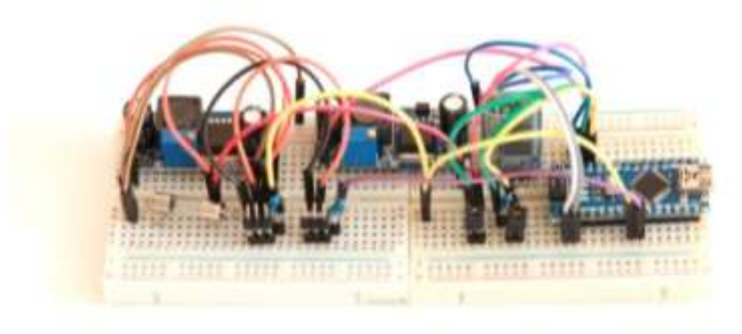
Instruments

Model name	Picture	Amount	Basic kit electronics	Pre-assembled kit
Crosshead screwdriver		1		X

Tweezers		1		X
Metal Long Needle Nose Plier Side Cutter Puzzle Modeling Precision Cutting		1		X

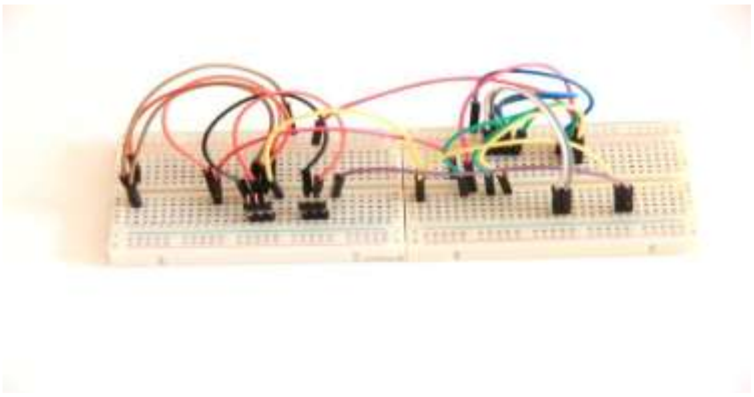


Set the items on the breadboard as indicated

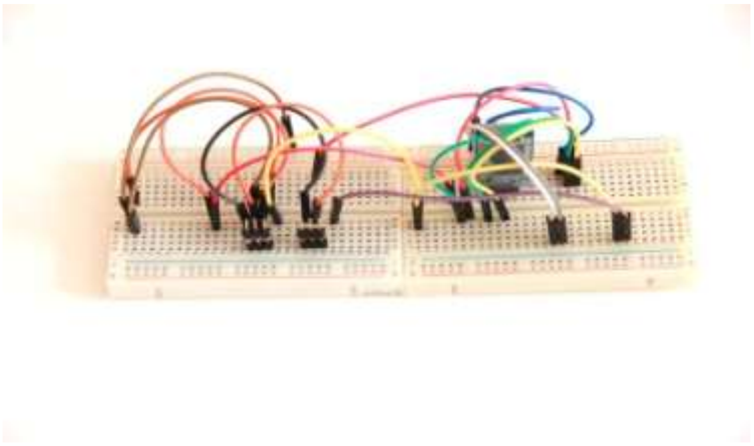


Connect the "ground" of all the items.

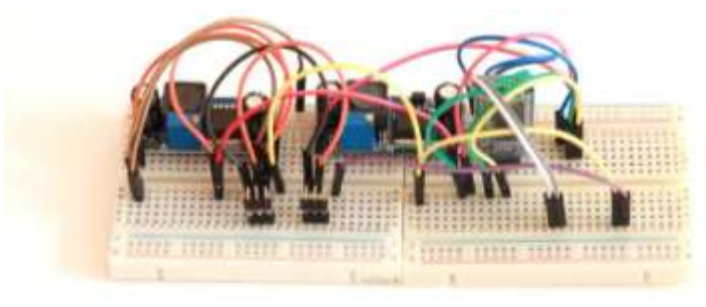
All the items are mounted so that one row on the contact breadboard builds a common "minus".



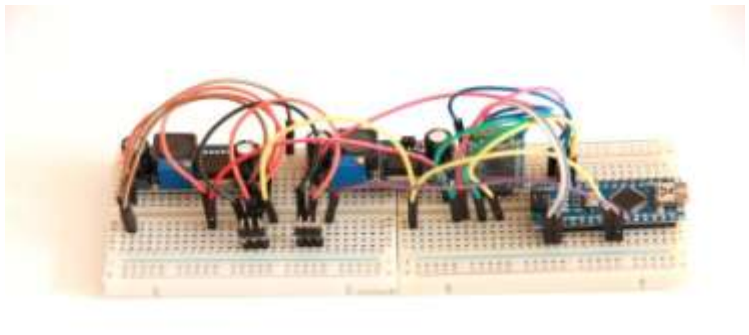
Connect the "minus" of the power connector, the "GRD" pins of the microcontroller and of the Bluetooth module to this "minus" with the black wires.



Connect the "ground" of the pins of the servo connectors to our common "minus" with the brown wires



Connecting the power converters



Power supply converters connection

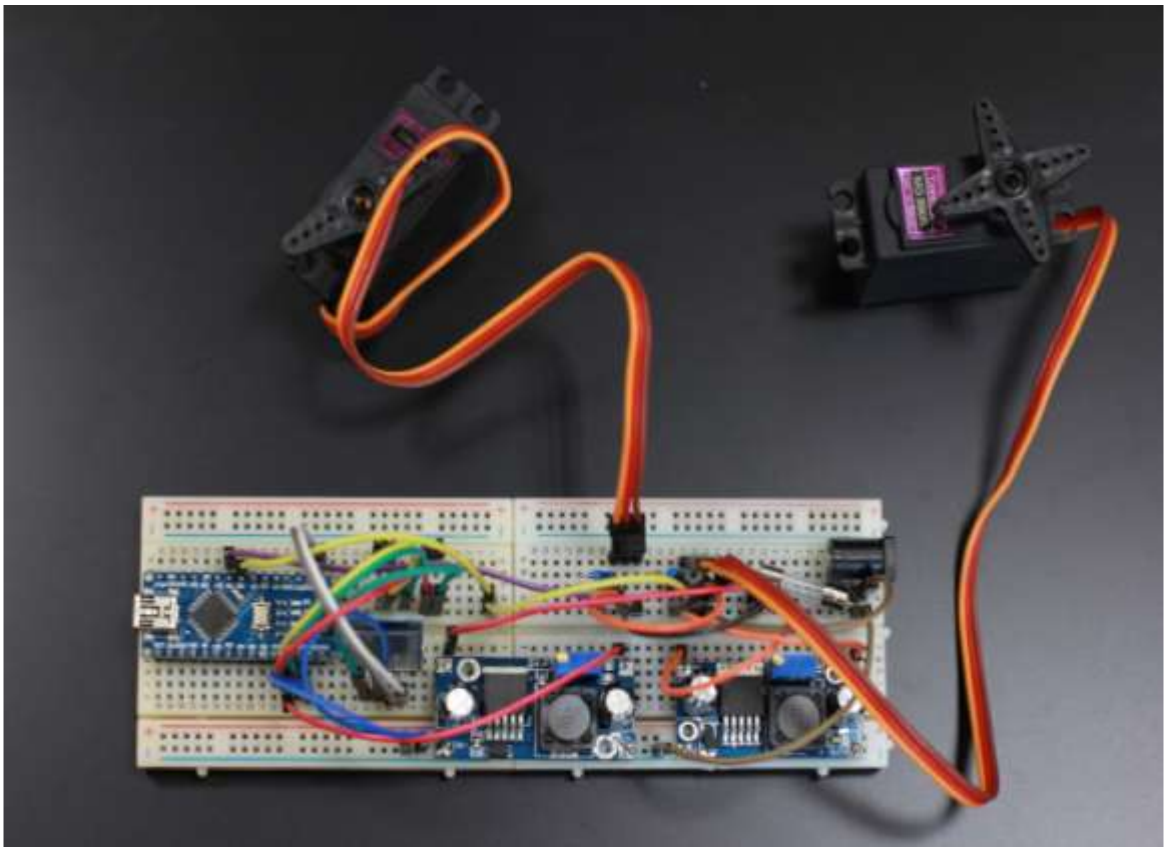
Using the red wires, connect the power connector "plus" and the DC / DC converters inputs through a fuse.

Using the orange wires, connect the output of the 7V converter to the "plus" pins of the servo connectors.

Using the yellow wires, connect the 5V converter output to the microcontroller and the jumper connectors in accordance with the schematics. Connect the "Vcc" pin of the bluetooth module to the connector of the corresponding jumper.

Connecting the "minuses" and "pluses" in this way, we join the assembled scheme through the power connector to the power source.

The microcontroller connection



Using the orange wire, connect the "REF" pin to the microcontroller power supply. This is an optional connection, but you may need it in the future when upgrading this version of the SelfieBot Base 1.1 electronics.

Using the green wire, connect the "RST" pin through the connector of the corresponding jumper to the microcontroller power supply through a 100 ohm resistor.

Connect the "Rx" pin of the Bluetooth module to the "Tx" pin of the microcontroller, and the "Tx" pin of the Bluetooth module to the "Rx" pin of the microcontroller.

Connect the servos to the microcontroller and install the jumpers.

Using the purple cable, connect the control pin of the connector for one of the servos through a 100 ohm resistor to the "D6" pin of the microcontroller.

Using the purple cable, connect the control pin of the connector for the other servo, through a 100 ohm resistor to the "D5" pin of the microcontroller. Since this connector is a bit away, use two blue wires connecting them via the contact breadboard.

The assembly of the SelfieBot Base 1.1 electronics on the contact breadboard is ready.

Brief instruction on the primary setup of the SelfieBot Base 1.1 electronics on the contact breadboard

Before supplying power to the assembled SelfieBot Base 1.1 electronics on the contact breadboard it is necessary to:

- ✓ check if every connection complies with the SelfieBot Base 1.1 schematics;
- ✓ check the setting of the voltage converters;

To check the setting of the voltage converters it is necessary to:

1) disconnect their outputs from the rest of the circuit - disconnect the yellow and orange wires from the "OUT +" pins of every DC converter;

2) connect the power supply to the power connector and turn it on;

3) make sure with a multimeter that the voltage converter for the servos (connected with the orange wire) supplies a voltage of 7V, otherwise set the voltage of 7V at the converter output by way of turning its potentiometer;

4) make sure with a multimeter that the voltage converter for the electronics connected with the yellow wire) supplies a voltage of 5V, otherwise set the voltage of 5V at the converter output by way of turningits potentiometer;

5) turn off the power supply and disconnect the electronics from the mains;

6) reconnect the outputs of all the DC converters to the electronics circuit by way of connecting the orange wire to the "OUT +" pin of the DC converter producing the voltage of 7V, and the yellow wire to the_“OUT+” DC pin of the converter producing the voltage of 5V.

Brief instruction on programming the microcontroller SelfieBot Base 1.1 electronics on the contact breadboard

For programming the microcontroller it is necessary to:

1) turn off and disconnect, if connected, the power supply from the power supply connector of the electronics;

2) remove the jumpers;

3) connect the microcontroller to the PC via a USB-miniUSB cable;

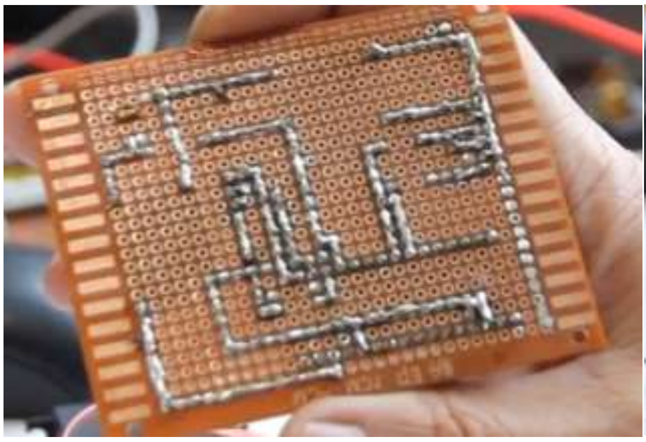
4) program (download the firmware)the microcontroller using Arduino IDE;

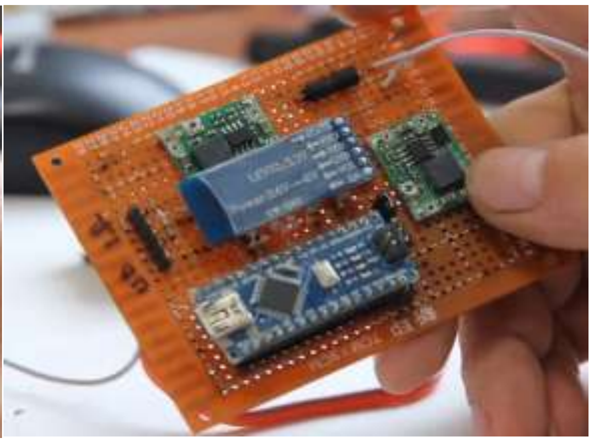
5) disconnect the USB-miniUSB cable from the microcontroller;

6) install the jumpers;

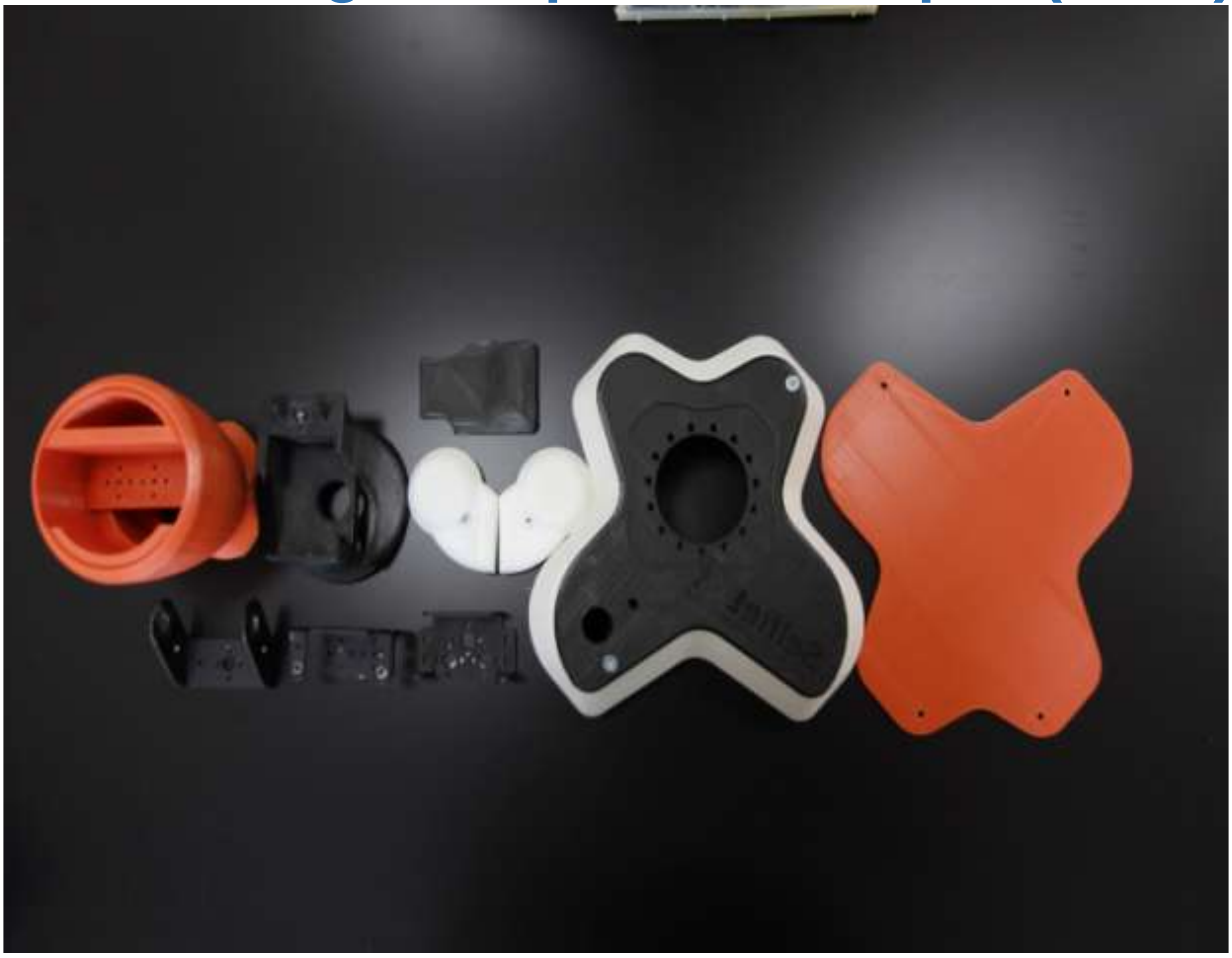
7) connect the power supply to the power supply connector of the electronics.

Make a PCB after you created working prototype on a breadboard**:**





Assembling 3D printed shape (mold)



Download the SelfieBot body files at this link: https://pinshape.com/items/21195-3d-printed-SelfieBot-by-endurance or http://www.thingiverse.com/thing:1508489

Use Cura (download Cura) to convert the SelfieBot body STL files into GCode for 3D printing.

Launch Cura app. and open the downloaded STL files one by one pressing the LOAD button in the upper left corner of the model view window until you fill the printing area.

You may use your usual 3D printer settings for SelfieBot body printing. Preferred printing settings for Wanhao Duplicator i3 with Nozzle size of 0.3 mm using PLA plastic in Cura are:

Quality:

- ✓ Layer height (mm) - 0.2
- ✓ Shell thickness (mm) - 0.9
- ✓ Retraction is enabled

Fill:

- ✓ Bottom/Top thickness (mm) - 1

- ✓ Fill Density (%) - 20-35 (any of this values are usual)

Speed and Temperature:

- ✓ Print speed (mm/s) - 30
- ✓ Printing temperature (C) - 210

Support:

- ✓ Support type - Touching buildplate
- ✓ Platform adhesion type - Raft

Filament:

- ✓ Diameter (mm) - 1.75
- ✓ Flow (%) - 100.0

Retraction:

- ✓ Speed (mm/s) - 40.0
- ✓ Distance (mm) - 1.5

In the menu FILE press “Save gcode as…” on a flash memory device compatible with your 3D printer (microSD for example).

Insert the flash memory card into a 3D printer, and print the files according to your 3D printer instructions.

Assembling process

When the electronics is ready, connect the servo motors to it and turn it on to set the servo motors in a zero position. Then turn it off and disconnect the servo motors.

Prepare the crosshead screwdriver, tweezers and pliers.

Prepare 26 screw nuts 3 mm in diameter and 7 screw nuts 4 mm in diameter.

Prepare countersunk head screws 3 mm in diameter:

10 mm - 8 pcs.

16 mm - 8 pcs.

round head screws 3 mm in diameter:

8 mm - 2 pcs.

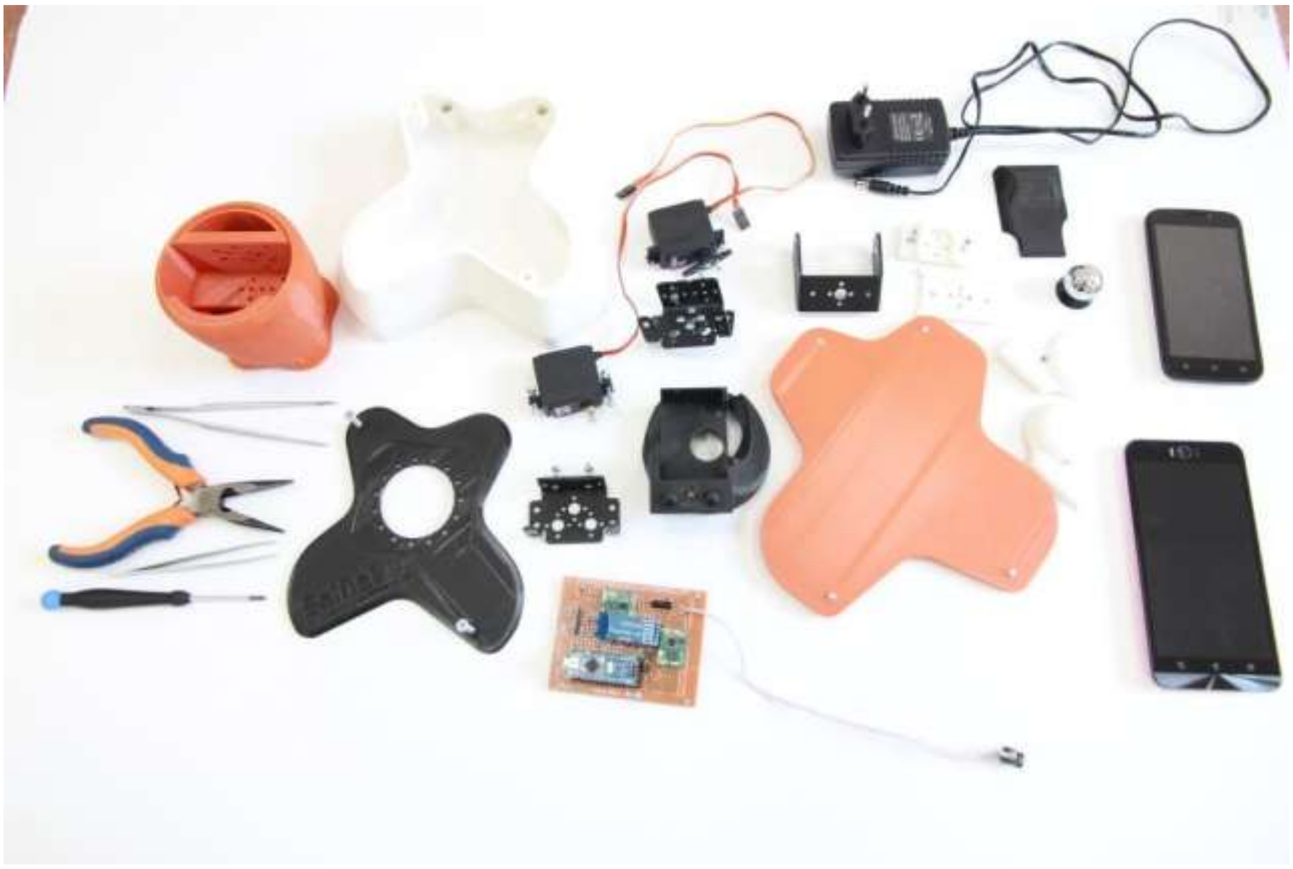
16 mm - 6 pcs.

20 mm - 10 pcs.

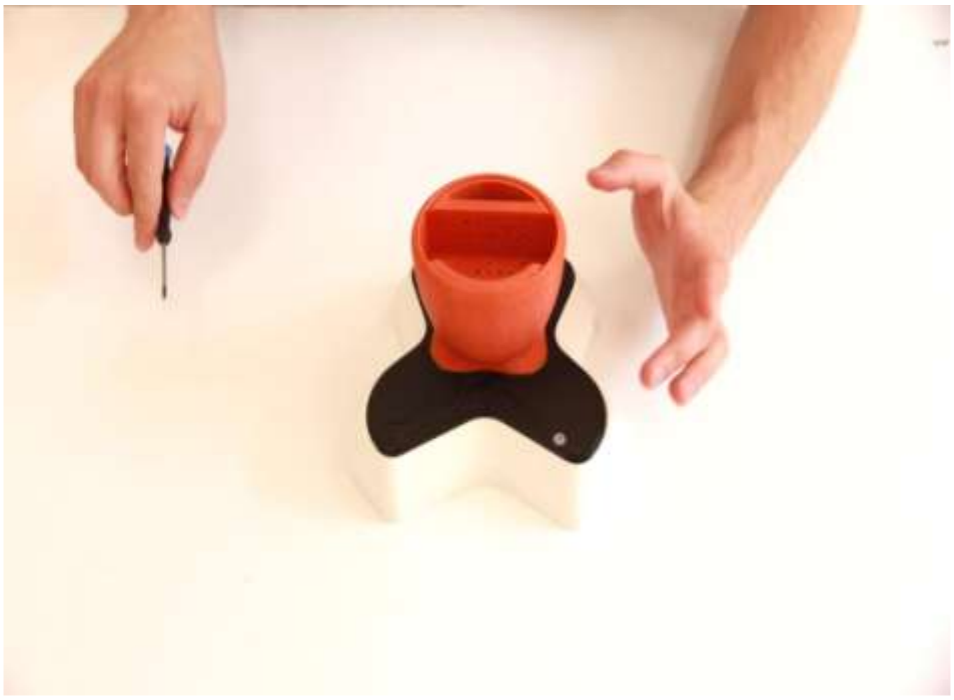
countersunk head screws 4 mm in diameter:

25 mm - 6 pcs.

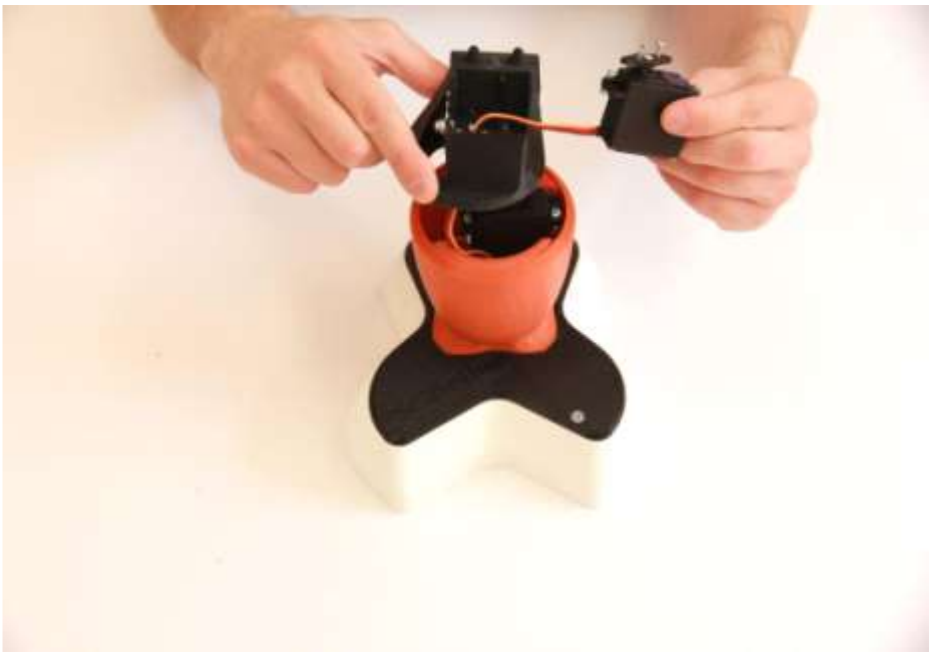
(you may need more screws - it is up to you).



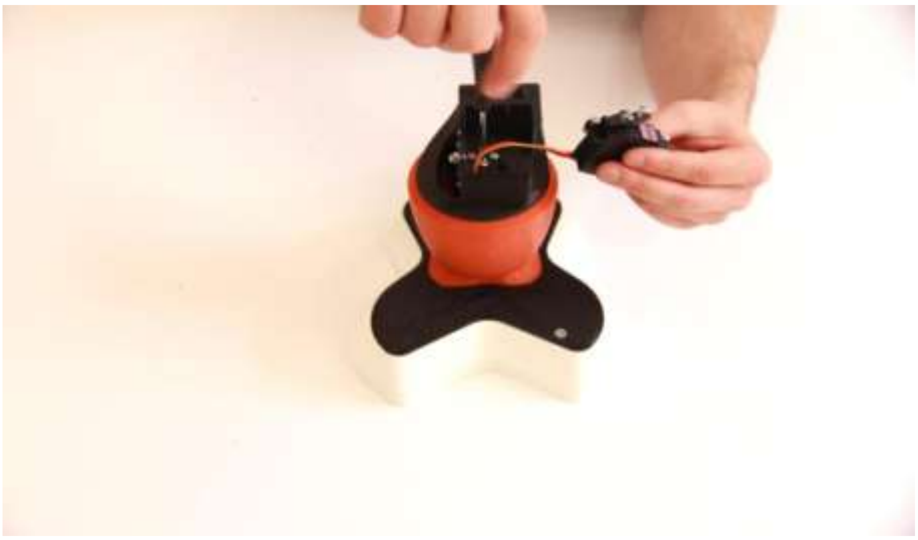
Screw part #1 with part #2 using two 4 mm screws and two screwnuts.



Screw parts #1 and #2 with part #3 using, at least, four screws and screw nuts 3 mm in diameter.

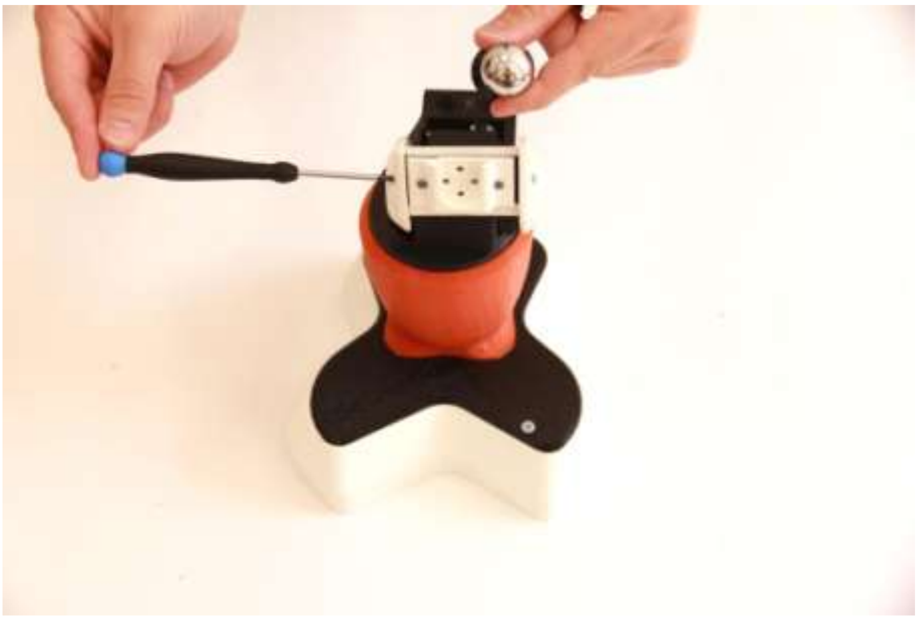


Screw the frame for a servo motor using two screws and screw nuts 3 mm in diameter.



Screw the servo motor with the frame with its shaft to the up in the center of part #3. Make sure that the servo cord is located in the low compartment.

Mount the round cap on the servo motor shaft and screw it with a 3 mm screw.



Mount part #4 with a second frame for a servo motor on it.

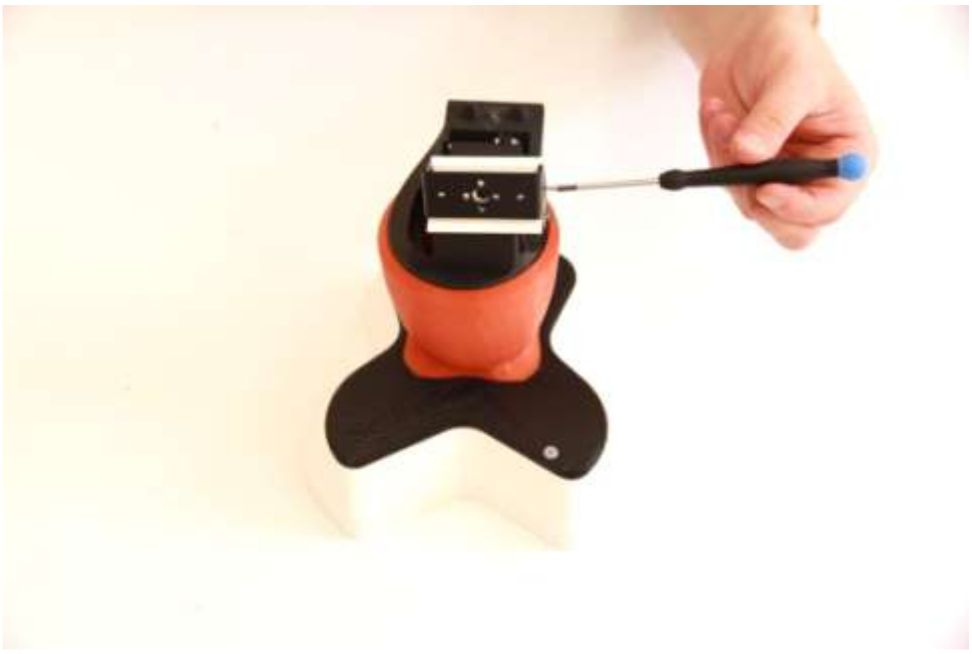
Screw the round cap of servo motor #1 using four countersunk head screws through the second frame.

Place the second servo motor cord in the low compartment and screw it to the frame.

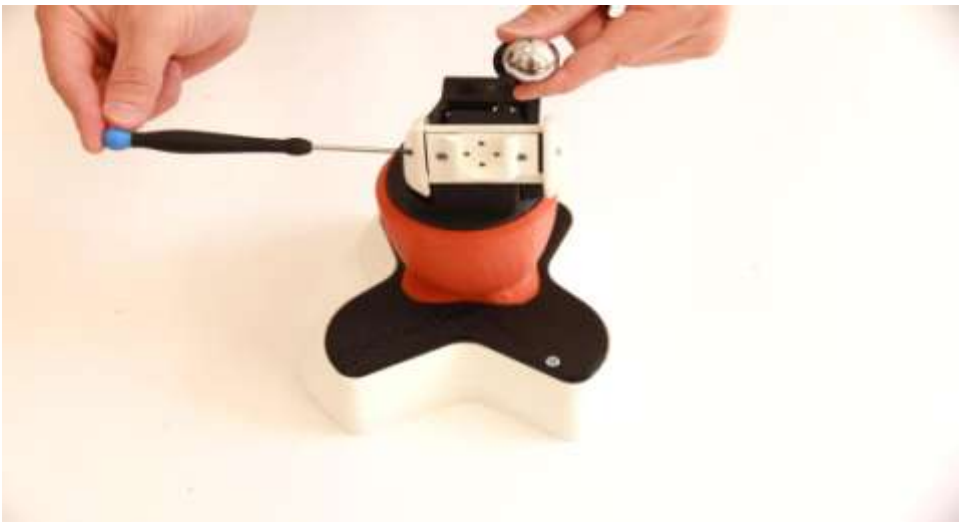
Screw a 3 mm round head screw through 4 mm screw nut to the frame at the servo motor shaft level from the other side.

Mount the servo motor shaft cap and then mount the bracket on the shaft from the one side and to the 3mm screw with 4 mm screw nut from the other.

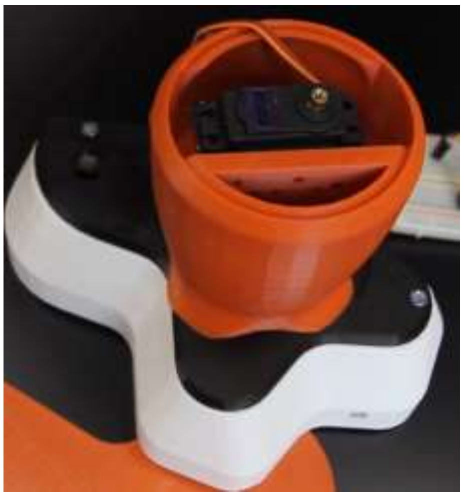
Screw the bracket to the servo motor shaft cap.



Mount parts #5 from both sides of the bracket and screw it.



Mount the electronics and connect the servo motors to it.



Screw bottom #6 with 4 mm screws and screw nuts.

Mount part #7.

Software Description SelfieBot Firmware Base 1.1

http://endurancerobots.com/azbnmaterial/software-description-work-selfie-bot-firmware-base-1-1/

```
/////////////////////////////////////////////////////////////////////////////////////////////
// SelfieBot Base © AlSHex
// 1.0
//
// Create: 27/03/2016
// Modification: 27/03/2016
//
// Description: Software for SelfieBot control. Electronics version Base 1.0
/////////////////////////////////////////////////////////////////////////////////////////////



#include <Servo.h>


// peripherals connecting
const int Servo1 = 5; // servo 1 [PWM]
const int Servo2 = 6; // servo 2 [PWM]
const int LED_tech = 13; // technological LED [RU: технологический светодиод]


// declaration and description of modules-functions, pinout and configuration
// ================================
// ========== Interface ==========
// ================================
const int UART_Speed = 9600; //UART speed

void _Interf_UART(unsigned int RST, unsigned int *Data);
// RST - reset: 0= normal functioning; 1= reset
// Data - command data, control byte array, that the module changes directly in memory
//
// Connecting to UART occurs via "Serial" Arduino library
// ====================================
// ========== Control Servo ==========
// ====================================
Servo servo1;
Servo servo2;

void _Control_Servo1(unsigned int Rst, unsigned int Rst_mode, unsigned int Ctrl, unsigned int Mode); //
servo for horizontal rotation
void _Control_Servo2(unsigned int Rst, unsigned int Rst_mode, unsigned int Ctrl, unsigned int Mode); //
servo for vertical rotation
// external control
// RST - reset: 0= normal functioning ; 1= reset
// Rst_mode - servo installation type in predetermined position (default): 0 = fast installation (sends the
default angle to servo); 1 = smooth installation (adjustable angle is growing slowly at the rate specified in
_Control_Servo module parameter)
// Ctrl - servo control data (command, the execution of which depends on the mode of operation Mode)
// Mode - mode: 0 = execution of the command considering timer command execution
//
// Internal settings
// - command values to change (increase or decrease) the angles of rotation and to stop rotation
const byte Servo1_Cmd_incr = byte('A'); // increasing angle of rotation
const byte Servo1_Cmd_decr = byte('D'); // decreasing angle of rotation
const byte Servo1_Cmd_stop = byte('x'); // stops rotation
const byte Servo1_Cmd_def = byte('R'); // command to setup default angle Pg_default const byte
Servo2_Cmd_incr = byte('W'); // increasing angle of rotation
const byte Servo2_Cmd_decr = byte('S'); // decreasing angle of rotation
const byte Servo2_Cmd_stop = Servo1_Cmd_stop; // stops rotation
const byte Servo2_Cmd_def = Servo1_Cmd_def; // command to setup default angle Pg_default
const int Servo1_Deg_default = 90; // угол поворота по умолчанию
const int Servo1_Deg_min = 0+5; // предельный минимальный угол поворота
const int Servo1_Deg_max = 180-5; // предельный максимальный угол поворота
const int Servo2_Deg_default = Servo1_Deg_default; // угол поворота по умолчанию
const int Servo2_Deg_min = Servo1_Deg_min; // предельный минимальный угол поворота
const int Servo2_Deg_max = Servo1_Deg_max; // предельный максимальный угол поворота
// - скорость поворота: неблокирующая задержка между изменениями углов на 1 градус при
повороте, если =0 - задержка не вносится [мс, >=0]
const long Servo1_Time_rotate_deg1 = 0;
const long Servo2_Time_rotate_deg1 = Servo1_Time_rotate_deg1;
// - регулятор скорости поворота при сбросе: блокирующая задержка, т.е. пока не будет установлен
заданный угол модуль не даст работать остальной части программы [мс, >0]
const int Servo1_Time_rst = Servo1_Time_rotate_deg1;
const int Servo2_Time_rst = Servo1_Time_rotate_deg1;
// - время актуальности команды на поворот (при Mode=0): после получения однократной команды
поворот будет остановлен через заданное время или по получению другой команды [мс, >0]
const long Servo1_Time_rotate = 2;
const long Servo2_Time_rotate = Servo1_Time_rotate;


// ==============================
// ========== MAIN ==============
// ==============================
unsigned int CMD[1] = {0}; // данные от интерфейса, 1 байт



void setup() {

  // первичная настройка доступных пинов
  pinMode(Servo1, OUTPUT); analogWrite(Servo1, 0);
  pinMode(Servo2, OUTPUT); analogWrite(Servo2, 0);
  pinMode(LED_tech, OUTPUT); digitalWrite(LED_tech, LOW);


  // начальная инициализация модулей
  Serial.begin(UART_Speed);
  servo1.attach(Servo1);
  servo2.attach(Servo2);

  digitalWrite(LED_tech, HIGH); // индикация инициализации - начало
  _Control_Servo1(1, 1, CMD[0], 0); // установка сервоприводов в положение по умолчанию
  __Control_Servo2(1, 1, CMD[0], 0);
  delay(1000);
  digitalWrite(LED_tech, LOW); // индикация инициализации - конец

}



void loop() {

  // опрос интерфейса, результат будет доступен в CMD
  __Interf_UART(0, CMD);

  // выполнение схем управления
  __Control_Servo1(0, 1, CMD[0], 0);
  __Control_Servo2(0, 1, CMD[0], 0);

}



// ========== Interface module ==========
void __Interf_UART(unsigned int Rst, unsigned int *Data) {

unsigned int data_uart;
static unsigned int cnt_byte;

  if (Rst == 0) {
    if (Serial.available() != 0) {
      data_uart = Serial.read();

      switch (cnt_byte) { // проверка целостности пакета, если хоть один сбой - сбрасываем приём и начинаем поиск заголовка нового пакета
        case 0:
          if (data_uart == byte('c')) { cnt_byte++; } else { cnt_byte = 0; }
          Data = 0;
          break;

        case 1:
          if (data_uart == byte('o')) { cnt_byte++; } else { cnt_byte = 0; }
          Data = 0;
          break;

        case 2:
          if (data_uart == byte('m')) { cnt_byte++; } else { cnt_byte = 0; }
          Data = 0;
          break;

        case 3:
          if (data_uart == byte('=')) { cnt_byte++; } else { cnt_byte = 0; }
          Data = 0;
          break;

        case 4: // если дошли до конца, то пакет верен и можно выдать команду
          cnt_byte = 0;
          Data[0] = data_uart;
          break;
      default:
        cnt_byte = 0;
        break;
    }
  } else {
    Data[0] = 0;
  }
 } else {
  cnt_byte = 0;
  Data[0] = 0;
 }
}



// ========== Control Servo1 module ==========
void _Control_Servo1(unsigned int Rst, unsigned int Rst_mode, unsigned int Cmd, unsigned int Mode) {
// Внешнее управление
// Rst - сброс: 0= нормальная работа; 1= сброс
// Rst_mode - тип установки сервопривода в заданное положение по умолчанию: 0=быстрая установка (на сервопривод выдаёт угол по умолчанию); 1= плавная установка (устанавливаемый угол нарастает медленно со скоростью, заданной в параметре модуля _Control_Servo)
// Ctrl - данные управления сервоприводами (команда, выполенние которой зависит от режима работы Mode)
// Mode - режим управления: 0= выполнение команды с учётом таймера времени выполнения команды

// Внутренние настройки
// - значения команд для изменения (увеличения или уменьшения) углов поворота и остановки поворота
const byte Cmd_incr = Servo1_Cmd_incr; // команда увеличения угла поворота
const byte Cmd_decr = Servo1_Cmd_decr; // команда уменьшения угла поворота
const byte Cmd_stop = Servo1_Cmd_stop; // команда остановка поворота
const byte Cmd_def = Servo1_Cmd_def; // команда установки угла по умолчанию Deg_default
// - максимальные и минимальные значения углов, угол поворота выставляемый по умолчанию при сбросе модуля
const int Deg_default = Servo1_Deg_default; // угол поворота по умолчанию
const int Deg_min = Servo1_Deg_min; // предельный минимальный угол поворота
const int Deg_max = Servo1_Deg_max; // предельный максимальный угол поворота
// - скорость поворота: неблокирующая задержка между изменениями углов на 1 градус при повороте, если =0 - задержка не вносится [мс, >=0]
const long Time_rotate_deg1 = Servo1_Time_rotate_deg1;
// - регулятор скорости поворота при сбросе: блокирующая задержка, т.е. пока не будет установлен заданный угол модуль не даст работать остальной части программы [мс, >0]
const int Time_rst = Servo1_Time_rst;
// - время актуальности команды на поворот (при Mode=0): после получения однократной команды поворот будет остановлен через заданное время или по получению другой команды [мс, >0]
const long Time_rotate = Servo1_Time_rotate;

static signed int deg; // угол поворота на сервопривод
static unsigned int incr, decr; // управляющие сигналы для алгоритма совершения поворота
static unsigned long time_cmd_detect, time_rotate_detect = 0; // значение времени получения команды и времени последнего изменёния угла поворота

 if (Rst == 0 & Cmd != Cmd_def) {

  if (Mode == 0) { // выполнение команды с учётом таймера времени выполнения команды
    if (Cmd == Cmd_incr) {
    incr = 1;
    decr = 0;
    time_cmd_detect = millis();
    } else if (Cmd == Cmd_decr) {
    incr = 0;
    decr = 1;
    time_cmd_detect = millis();
    } else if (Cmd == Cmd_stop) {
    incr = 0;
    decr = 0;
    }

    if (millis() - time_cmd_detect > Time_rotate) {
    incr = 0;
    decr = 0;
    }
  }

  if (Time_rotate_deg1 == 0) { // алгоритм поворота
    if (incr == 1) {
      if (deg < Deg_max) { deg++; } // поворачиваем к 180
    } else if (decr == 1) {
      if (deg > Deg_min) { deg--; } // поворачиваем к 0
    }
    servo1.write(deg);
  } else {
    if (millis() - time_rotate_detect > Time_rotate_deg1) { // задержка при повороте - регулирование
скорости поворота
    if (incr == 1) {
      if (deg < Deg_max) { // поворачиваем к 180
        deg++;
        time_rotate_detect = millis();
      }
    } else if (decr == 1) { // поворачиваем к 0
      if (deg > Deg_min) {
        deg--;
        time_rotate_detect = millis();
      }
    }
    servo1.write(deg);
    }
  }

 } else {

  // установка по умолчанию
  while (deg != Deg_default) {
    if (Rst_mode == 0) {
    servo1.write(Deg_default);
    deg = Deg_default;
    } else {
    if (deg < Deg_default) { deg++; } else { deg--; }
    servo1.write(deg);
    delay(Time_rst);
    }
  }

  incr = 0;
    decr = 0;
  }
}


// ========== Control Servo2 module ==========
void _Control_Servo2(unsigned int Rst, unsigned int Rst_mode, unsigned int Cmd, unsigned int Mode) {
// Внешнее управление
// Rst - сброс: 0= нормальная работа; 1= сброс
// Rst_mode - тип установки сервопривода в заданное положение по умолчанию: 0=быстрая установка (на сервопривод выдаёт угол по умолчанию); 1= плавная установка (устанавливаемый угол нарастает медленно со скоростью, заданной в параметре модуля _Control_Servo)
// Ctrl - данные управления сервоприводами (команда, выполенние которой зависит от режима работы Mode)
// Mode - режим управления: 0= выполнение команды с учётом таймера времени выполнения команды

// Внутренние настройки
// - значения команд для изменения (увеличения или уменьшения) углов поворота и остановки поворота
const byte Cmd_incr = Servo2_Cmd_incr; // команда увеличения угла поворота
const byte Cmd_decr = Servo2_Cmd_decr; // команда уменьшения угла поворота
const byte Cmd_stop = Servo2_Cmd_stop; // каманда остановки поворота
const byte Cmd_def = Servo2_Cmd_def; // команда установки угла по умолчанию Deg_default
// - максимальные и минимальные значения углов, угол поворота выставляемый по умолчанию при сбросе модуля
const int Deg_default = Servo2_Deg_default; // угол поворота по умолчанию
const int Deg_min = Servo2_Deg_min; // предельный минимальный угол поворота
const int Deg_max = Servo2_Deg_max; // предельный максимальный угол поворота
// - скорость поворота: неблокирующая задержка между изменениями углов на 1 градус при повороте, если =0 - задержка не вносится [мс, >=0]
const long Time_rotate_deg1 = Servo2_Time_rotate_deg1;
// - регулятор скорости поворота при сбросе: блокирующая задержка, т.е. пока не будет установлен заданный угол модуль не даст работать остальной части программы [мс, >0]
const int Time_rst = Servo2_Time_rst;
// - время актуальности команды на поворот (при Mode=0): после получения однократной команды поворот будет остановлен через заданное время или по получению другой команды [мс, >0]
const long Time_rotate = Servo2_Time_rotate;

static unsigned int deg; // угол поворота на сервопривод
static unsigned int incr, decr; // управляющие сигналы для алгоритма совершения поворота
static unsigned long time_cmd_detect, time_rotate_detect = 0; // значение времени получения команды и времени последнего изменёния угла поворота

  if (Rst == 0 & Cmd != Cmd_def) {

    if (Mode == 0) { // выполнение команды с учётом таймера времени выполнения команды
      if (Cmd == Cmd_incr) {
        incr = 1;
        decr = 0;
        time_cmd_detect = millis();
      } else if (Cmd == Cmd_decr) {
        incr = 0;
        decr = 1;
        time_cmd_detect = millis();
      } else if (Cmd == Cmd_stop) {
        incr = 0;
        decr = 0;
      }

      if (millis() - time_cmd_detect > Time_rotate) {
        incr = 0;
        decr = 0;
      }
    }

    if (Time_rotate_deg1 == 0) { // алгоритм поворота
        if (incr == 1) {
          if (deg < Deg_max) { deg++; } // поворачиваем к 180
        } else if (decr == 1) {
          if (deg > Deg_min) { deg--; } // поворачиваем к 0
        }
        servo2.write(deg);
    } else {
      if (millis() - time_rotate_detect > Time_rotate_deg1) { // задержка при повороте - регулирование
скорости поворота
        if (incr == 1) {
          if (deg < Deg_max) { // поворачиваем к 180
            deg++;
            time_rotate_detect = millis();
          }
        } else if (decr == 1) { // поворачиваем к 0
          if (deg > Deg_min) {
            deg--;
            time_rotate_detect = millis();
          }
        }
        servo2.write(deg);
      }
    }

  } else {

    // установка по умолчанию
    while (deg != Deg_default) {
      if (Rst_mode == 0) {
        servo2.write(Deg_default);
        deg = Deg_default;
      } else {
        if (deg < Deg_default) { deg++; } else { deg--; }
        servo2.write(deg);
        delay(Time_rst);
      }
    }

    incr = 0;
    decr = 0;
  }
}
```

Purpose of the program

The software is intended to be loaded into the microcontroller hardware SelfieBot Hardware Base, built on the base board Arduino Nano, comprising a microcontroller Atmel ATmega168 or ATmega328.

This software makes it possible to connect to the device SelfieBot and control the connected servos.

Description of the program

After starting the device initialization is done: run the setup function commands — Initialization pins, UART configuration interface, setting the servo to the initial position. During initialization, the LED lights technology (Arduino Nano to the boards are usually marked with the letter L), which goes out on successful initialization after 1 second.

After initialization, the loop function starts running in an infinite loop — going survey software interface module _Interf_UART to transfer results to the software modules and _Control_Servo1 _Control_Servo2.

_Interf_UART Software module provides a UART reception for the team. In the received stream for UART is a search and identification of a data packet in accordance with a protocol for information exchange. In the case of command detection occurs issuance of the control byte. Work with internal UART provides a library of Serial.

Since the protocol is fixed, then, to the stability of the failure, the module receives each new byte, which came on the UART, and compares it with the expected, in the case of coincidence of the structure of the package is considered to be the last byte of the command and issued. If the bytes do not expect the reception of reset. In the event of failure of the whole exchange, this ensures the correct reception of the next packet (following the flawed) without re-initialization reception timeout circuitry that determines the end of data reception.

Software modules and Control_Servo1 _Control_Servo2 ensure alignment to the servos using the Servo Library command. Upon receipt of the control byte of the interface module takes place identification byte management and development of signals to permit rotation of the actuator in the desired direction. After a certain time (Time_rotate) stops rotation, if not received any other team identified.

Information exchange protocol

Interaction with external devices takes place via UART at 9600 baud.

The data packet consists of five bytes. The first four bytes represent a prefix and are constant in each package. The fifth byte is a byte command.

The structure of the package

№ Byte	1	2	3	4	5
Command symbol	c	o	m	=	

Command byte	0x63	0x6F	0x36	0x3D	

Management Team (fifth byte in the packet)

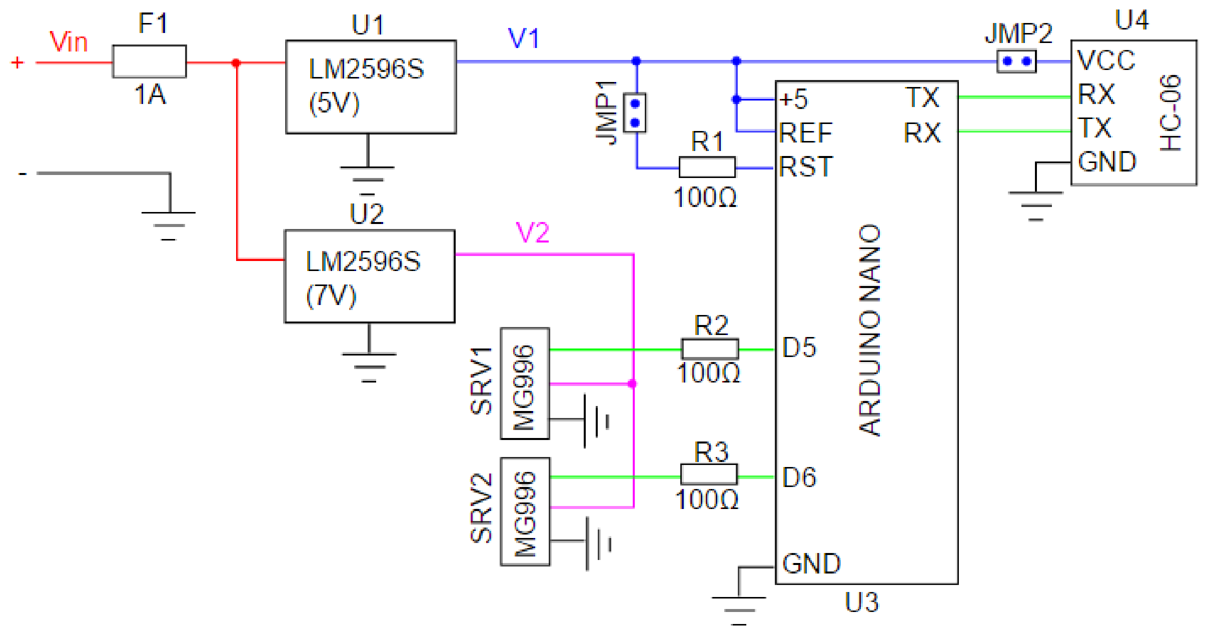
Command symbol	Command byte	Command
W	0x57	Rotation of a vertical servo to the up
S	0x53	Rotation of a vertical servo to the down
A	0x41	Rotate of a horizontal servo to the left
D	0x44	Rotate of a horizontal servo to the right
x	0x78	Stop of any movements
R	0x52	Return servos to the starting point

Electronic circuit SelfieBot Hardware Base 1.1 description

Scheme description

The electronic circuit is designed to control the device SelfieBot. This scheme allows you to communicate with the device via Bluetooth and control two servos. To control device intended SelfieBot Droid program. Thus, SelfieBot Firmware software must be installed on the device microcontroller Base.

Schematic diagram



Vin = 12 VDC	F1 = FUSE, 1A	U1 = Module DC/DC LM2596S	JMP1 = JUMPER
V1 = 5 VDC	R1 = 100Ω	U2 = Module DC/DC LM2596S	JMP2 = JUMPER
V2 = 7 VDC	R2 = 100Ω	U3 = Module ARDUINO NANO	SRV1 = MG996
	R3 = 100Ω	U4 = Module BLUETOOTH HC-06	SRC2 = MG996

Description of the scheme

The device is powered from an external power source voltage of 12V, able to give at least 1A. external power line protected by a fuse nominal value of 1A. DC-DC converters D1 and D2 are organizing two channels of power: the microcontroller and D3 Servo SRV1 and SRV2. Converters are adjusted so that the microcontroller and Bluetooth D4 D3 module is fed 5V, 7V to the servos served.

Microcontroller D3 is connected to the Bluetooth module for the D4 UART interface via the appropriate conclusions, available in both modules. Through the resistors R2 and R3 outputs D5 and D6 D3 of the microcontroller pins are connected to the control on the connectors, which are connected servos.

Login RST microcontroller D3 pulled up to the power supply line + 5V via a resistor R1 and a removable jumper JMP1. Bluetooth D4 module via removable peremychkuJMP2 connected to + 5V line. In normal operation and JMP1 JMP2 device jumper must remain in place. When programming the microcontroller and D3 JMP1 JMP2 jumper removed.

Download center

http://endurancerobots.com/azbnmaterial/download-center/

For your convenience you can find everything you need in our download center.

Software source, stl files for 3d printing, schematics.

The demo version of the program SelfieBot

Demo on the Android console for testing

STL files for the robot body 3D printing

Software for the SelfieBot control

Electronics design

SelfieBotDroid archive

Tech overview

Endurance robots teaches and explains how to make a nice, practical and fully 3D printed robot for education in classrooms to teach Computer Science, Science, Technology Engineering, and Mathematics (CS-STEM). We want to inspire more and more high school and college students to do DIY robotics things that is why we share our knowledge and experience.

To operate a DIY SelfieBot you will need:

1. Constant current source (12V)
2. A smartphone/tablet with Android version 4.2.2 and higher, Bluetooth, Wi-Fi modules or 3G / 4G.
3. Stable high bandwidth Internet connection (WiFi, 3G, 4G).

SelfieBot characteristic data:

1. Angles of horizontal rotation from -90 to 135 degrees.
2. Angles of vertical rotation from -30 to 90 degrees (the threshold values depend on the holder construction and the smartphone/ tablet size).
3. The «demo» mode operation time (with continuous rotation and the payload weight of 150 g) is 5h.
4. Allowed dimensions of the smartphone/tablet are 5 — 10 inches; recommended — 7 inches.
5. The device body is fully 3D printed and can be made out of ABS or PLA
6. Rotation commands transfer via Bluetooth.

SelfieBot modes of operation are selected in the Android app:

1. CONNECT OVER INTERNET / CONNECT OVER BLUETOOTH. Remote rotation control via Internet.
2. FACE TRACKING BOT. Centering of the person's face who is in sight of the smartphone/tablet front camera.
3. DEMO. Continuous rotation up to the threshold values of the rotation angles.

Operation, transportation, storage:

1. The device is intended for use indoors and outdoors under a cover at temperatures of 0 — 55 C.

2. Careful transportation due to the plastic parts.
3. Storage is possible at the temperatures of -20 to +70 C.

Deployment Instruction

1. Connect your SelfieBot power connector to the DC power supply.
2. Mount the smartphone / tablet holder on the SB until it clicks.
3. Secure a smartphone / tablet in the holder until it clicks.
4. Press the SB power button on the front part of the device body.

Enabling / activation of the operation mode.

1. Activate the Bluetooth (required) and the Internet connection (optional) on your smartphone / tablet (depending on the preferred mode of operation);
2. Activate the Droid app on your smartphone / tablet;
3. Push «Connect over Bluetooth» in the Android app. When connecting for the first time, the Bluetooth PIN is requested. Enter 1234;
4. When «Holder is connected» message will appear in the top left corner run a selected messenger (Skype, Viber, WhatsApp) on the smartphone / tablet.
5. When connecting a remote Android device to the SelfieBot, run on your Android device the Droid app and press the button «Connect over Internet»; four translucent arrow buttons to control robot rotation will appear on the touch screen. For video connection start a selected messenger on your Android device and make a video call to the account running on the SelfieBot smartphone / tablet;
6. If you intend to control the SB using a Windows device, it is enough to go to http://mikeravx.gear.host/SelfieBotWeb/, enter the device ID (987654321) and click «Connect «; then you can control the robot rotation with the buttons below.

Testing connection bluetooth.

With this method, it is necessary:

1) to install an Arduino Bluetooth Controller 1.2 (enclosed) on your phone and check the way it works with a certain phone and electronics. Call me and I'll tell you how to set it up;

2) to replace the Arduino Bluetooth Controller 1.2 with the SelfieBot Droid software and proof test it once more. If you have problems in the first step, look for its cause in the electronics or phone. If failure occurs in the second step, something is wrong with the SelfieBot Droid app. If no failure occurs in the normal mode of operation, run troubleshoot diagnostics of the SelfieBot Droid app. If failure occurs test it using Arduino Bluetooth Controller 1.2. If there's no failure there's a problem with the SelfieBot Droid software. If there's failure then something is wrong with the electronics or phone. In this case send me your test program for additional checking.

Bluetooth connection testing

Date: 19-06-2016

In use:

- ✓ SelfieBot Base 1.1 (Bluetooth HC-06)

- ✓ 9B/1A power adapter (arrived on 16-06-2016 together with the problem-plagued SelfieBot 3.2)
- ✓ MG996R servos
- ✓ Motorola Droid Turbo (Android 5.1)
- ✓ Arduino Bluetooth Controller 1.2

Testing program:

17:00 – power supply connection

17:02 – activating, operating

17:07 – operation testing, re-activating, operating

17:20 - operation testing, shutdown

17:50 – activating, operating

18:30 – operation testing, switching off

Conclusion:

- no troubles revealed