22. July 2019

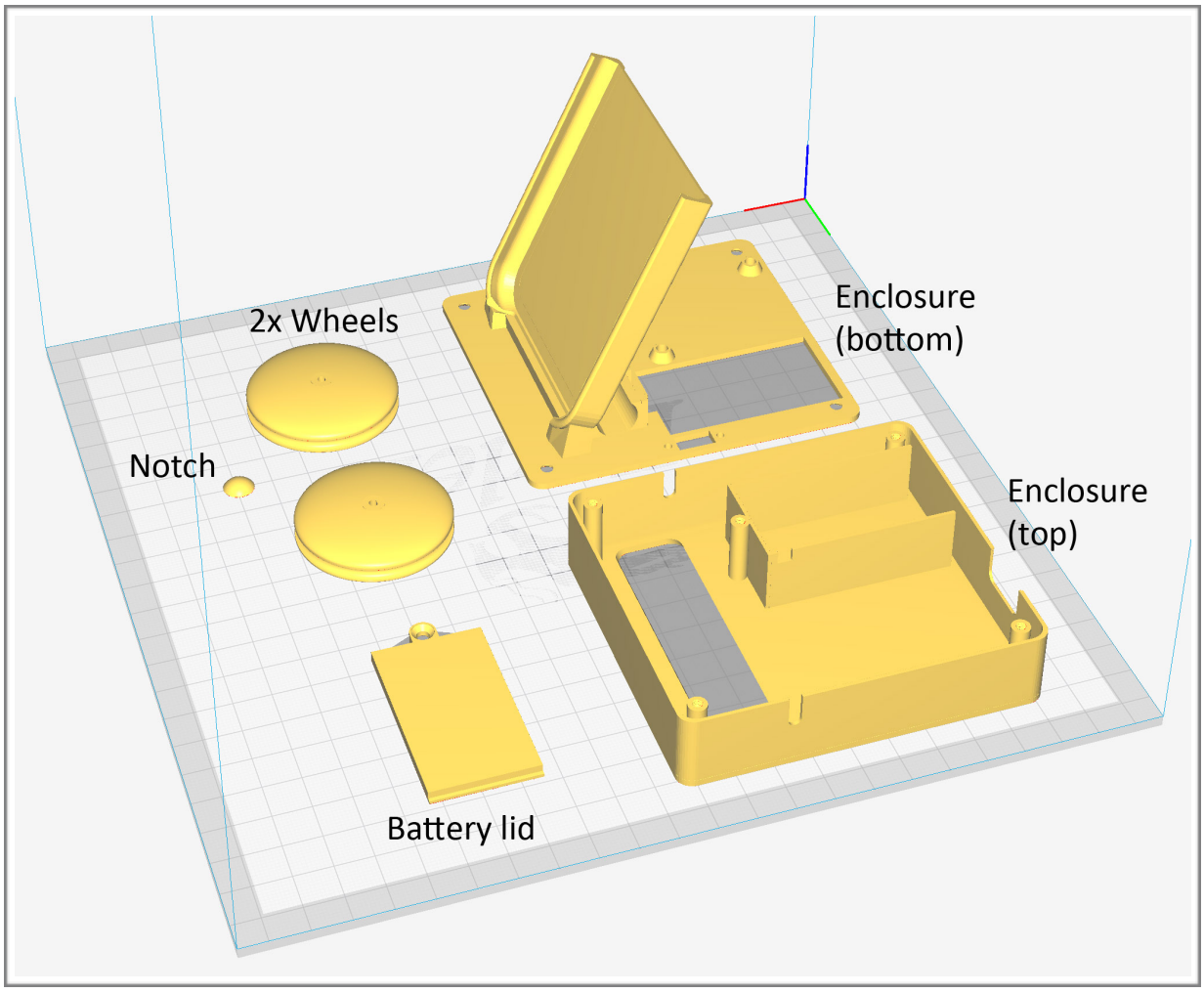
Kortexino Bot

Instruction manual

3D Printing and additional components

3D printing:

All components can be printed using PLA, 20% infill without supports.



Additional components:

8x M2.5 countersunk screws 11mm length or shorter

3x M2.5 nuts

2x M2 buttoned screws 8-10mm length

2x M2 nuts

2x silicone O-ring 45mmx38mmx3.5mm (for wheels)

2x N30 motor 300rpm DC 6V

1x SS-12F15G6 SPDT panel switch

1x 9V battery snap connector with plain wire ends

1x 9V battery

1x Romeo BLE mini

Wires and soldering equipment

Assembly

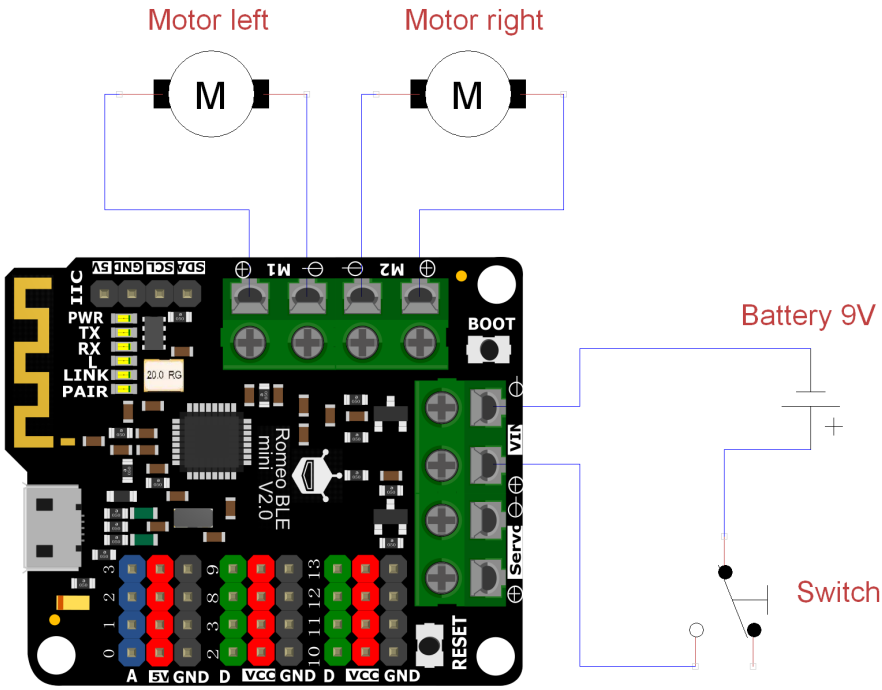
1. Mount the Romeo BLE mini board on the Enclosure (bottom) with 3 M2.5 screws and nuts. Do not over-tighten the screws as this could damage the 3D printed plastic parts.

2. Solder wires on the N30 motors and solder the + terminal of the 9V battery snap connector to the middle connector of the panel switch. Solder another wire to one of the other panel switch connectors.

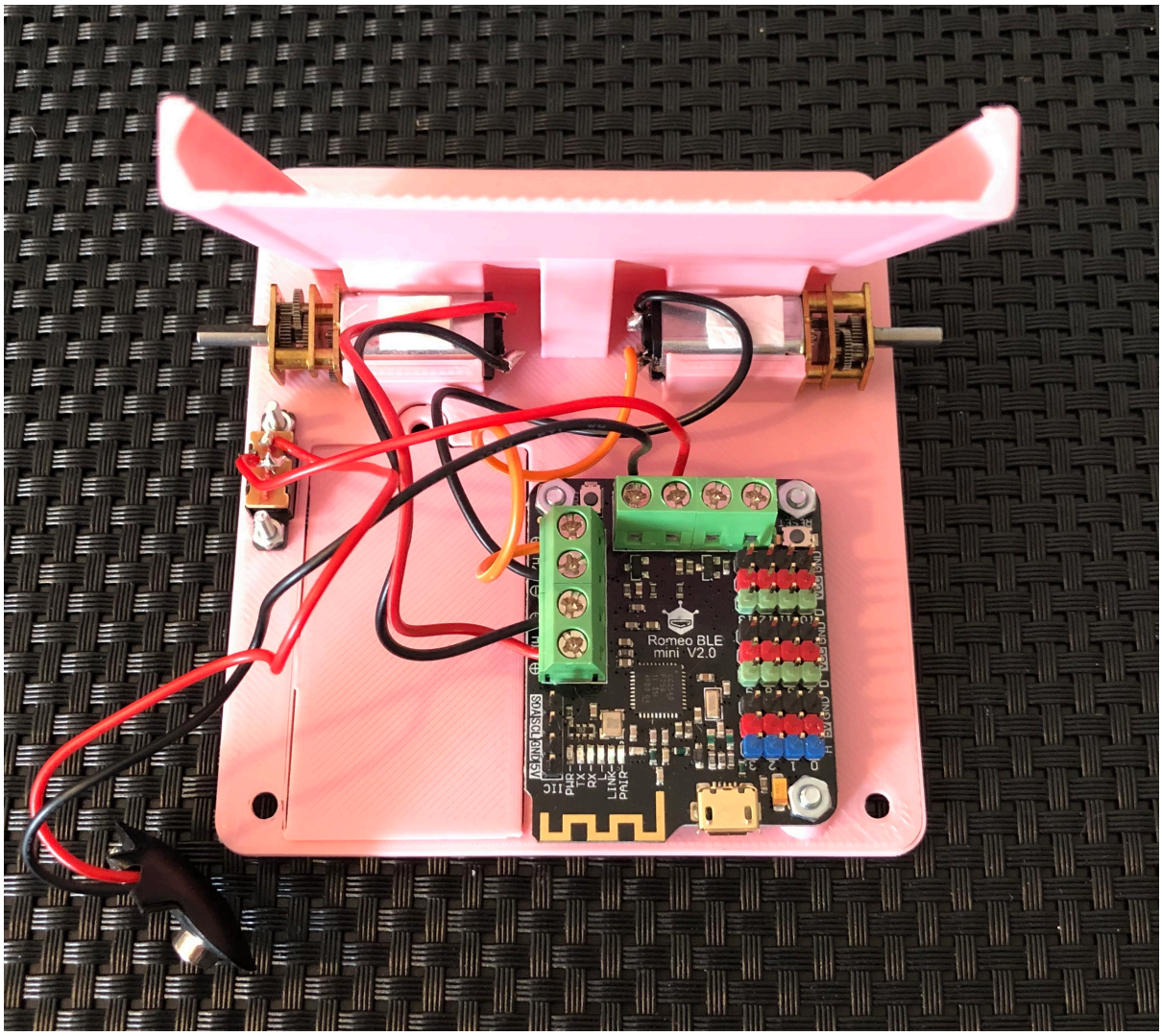
3. Carefully insert the N30 motors into the Enclosure (bottom) .

4. Mount the panel switch to the Enclosure (bottom) with 2 M2 screws and nuts. Do not over-tighten the screws.

5. Connect the wires to the Romeo BLE mini board. Make sure that you apply power with the correct polarity. Reverse polarity will damage the board. Also make sure to connect the motors with the correct polarity.



6. The Enclosure (bottom) should now look like this:



7. Slide the Enclosure (top) onto the Enclosure (bottom). Guide the wires of the 9V battery snap connector through the gap in the battery housing compartment of the Enclosure (top).

8. Use 4 M2.5 screws to firmly attach Enclosure (top) to Enclosure (bottom). Do not over-tighten the screws.

9. Connect the 9V battery.

10. To close the battery compartment, insert the battery lid into Enclosure (bottom). Make sure to insert the lid in the correct orientation, with the indentation for the countersunk screws facing outwards. The lid also has a small wedge that locks it into the enclosure on the side opposite to the fixation screw. Secure the lid with a M2.5 screw.

11. Glue the notch to the Enclosure (bottom) as shown in this picture.

12. Attach the silicone O-rings to the wheels and attach the wheels to the N30 motors. The assembly is now finished.

Programming

1. Install the Arduino IDE for MacOS or Windows and download the example Kortexino code.
2. Connect Kortexino to your computer via a USB cable. You do not need to switch on battery power. Power provided via USB is sufficient.
3. If necessary, install drivers for the Romeo BLE mini board.
4. In the Arduino IDE, load the "Shybot" example Kortexino code.
5. In Tools/Board, choose "Arduino/Genuino Uno".
6. In Tools/Port, choose the port that is linked to "Arduino/Genuino Uno".
7. Hit the upload button. After successful upload, disconnect the USB cable and switch on the Battery switch.
8. Start the Kortexino app.
9. Hit the settings button on the top right of the Kortexino app screen and choose “Connect Romeo BLE”, then “Back". You only have to do this once, as long as you do not wish to connect to different BLE devices with the Kortexino app.
10. Choose one of the animals or brain mode.
11. Warning: As soon as the Kortexino app connects to Kortexino bot, the motors might start to move and the Kortexino bot might behave unexpectedly. Do not insert your iPhone into the Kortexino bot before you make sure that the robot is in a safe environment.

Also secure your Kortexino bot and make sure that it does not damage itself by running into obstacles or falling off edges before you proceed. It is not recommended to use Kortexino on elevated surfaces, such as tables.

12. Tap the bluetooth connect button on the top right of the Kortexino app screen. The app should now scan for the Kortexino bot and then connect to it.

13. Carefully observe, how the Kortexino bot reacts to your face position relative to your iPhone. Once you made sure that the movements are safe, carefully insert the iPhone into Kortexino bot.

14. If the robot does not behave as expected, i.e. does not turn away from your face as expected for the uploaded shybot code, but instead moves forward or backward, it is likely that one of the motors was connected with incorrect polarity. In the more complex starebot example code, if the robot moves in the opposite direction of your face, either both motors were connected with incorrect polarity, or the right and left motor connections are inverted.

A simple Kortexino program explained

Lets start with a simple code example, the “Shybot”:

```
#include "Kortexino.h"
#include "Robot.h"

Kortexino kortexino;
Robot robot;

void setup()
{
  robot.start();
}

void loop()
{
  kortexino.read_data();
  if(kortexino.data_available)
  {
    if (kortexino.face_detected == true)
    {
      if(kortexino.yaw_to_bot > 1)
        robot.turn_right(20);
      else
        robot.turn_left(20);
    }
    else
      robot.stop_movement();
  }
}
```

This code generates a simple behaviour. If the Kortexino app detects a face it will turn the Kortexino bot away from the face. To do this more efficiently, the app will check if the face is located to the right or left of

the iPhone, and then will turn the Kortexino bot in the direction that points away from the face quicker.

The first 4 lines are always the same for a Kortexino program:

```
#include "Kortexino.h"
#include "Robot.h"

Kortexino kortexino;
Robot robot;
```

These first 2 lines load additional code that deals with the hardware specifics of the Romeo BLE micro controller and the communication protocol of the Kortexino app. To control your robot you do not need to understand these specifics, however, if you are interested, you can open the code and inspect how it is working (it is actually fairly simple). Make sure that the files Kortexino.h, Kortexino.cpp, Robot.h and Robot.cpp are present in the folder that contains your code. The downloaded example code already contains these files.

The next 2 lines generate two objects that you can then use to receive the Kortexino communication and to control the robot movements.

The next 4 lines contain code that is only executed one time at the start of the program. This code is framed by curved brackets in a special function called setup. You can set certain values within these brackets that need to be set at the beginning of your code. Here, the only thing that happens is that the Romeo BLE micro controller is started, which always has to happen in the setup function of a Kortexino program.

```
void setup()
{
  robot.start();
}
```

The next part contains all the code for the specific behaviour of the Shybot program. It is inside a function called loop that again is framed by curved brackets. This function will be restarted in an endless loop each time is finishes.

```
void loop()
{
  kortexino.read_data();
  if(kortexino.data_available)
  {
    if (kortexino.face_detected == true)
    {
      if(kortexino.yaw_to_bot > 1)
        robot.turn_right(20);
      else
        robot.turn_left(20);
    }
    else
      robot.stop_movement();
  }
}
```

kortexino.read_data():

First, the function read_data() is called, which is included in the kortexino object. This makes sure that new data obtained via bluetooth is read into the kortexino object.

if(kortexino.data_available)

Then, the program checks, if bluetooth data was indeed read. If this is the case, then the code inside the curved brackets is executed.

if(kortexino.face_detected == true)

Here, the program checks, if the Kortexino app detected a face. If this is the case, then the code inside the curved brackets is executed. If not, then the code following the else statement is executed.

if(kortexino.yaw_to_bot > 0)

Now, the program checks the yaw angle (left vs right facing the iPhone). If the face is positioned to the left of the iPhone, then this value is positive, if it is positioned to the right, it is negative. If it is positive, the following code is executed:

robot.turn_right(20);

This makes the two motors of the Kortexino bot turn in opposite direction with the speed of 20, leading to a turning movement to the right. The alternative code robot.turn_left(20); is executed if the face was positioned on the opposite side of the iPhone.

Robot.stop_movement();

This code is below the else statement that belongs to the code if(kortexino.face_detected == true). Therefore, this code will be executed if no face is detected. It stops the movement of both motors and therefore stops any turning or forward/backward movements.

Together, this code enables the behaviour of shybot, which will always look away from a face that it detects.

Programming Reference

The Kortexino Object:

Variables:

bool kortexino.data_available

True, if data was received via Bluetooth

bool kortexino.face_detected

True, if a face is currently being detected

byte kortexino.face_distance

The distance between the detected face and the iPhone in cm

int kortexino.yaw_to_bot

The yaw angle (i.e. left vs right) between the detected face and the iPhone in positive or negative degrees

int kortexino.pitch_to_bot

The pitch angle (i.e. up vs down) between the detected face and the iPhone in positive or negative degrees

int kortexino.roll_to_bot

The roll angle (i.e. face roll clockwise vs anticlockwise) between the detected face and the iPhone in positive or negative degrees

byte kortexino.smile

Detected smile face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.tongue_out

Detected tongue out face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.kiss

Detected kissing lips face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.mouth_open

Detected open mouth face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.wink_left

Detected wink of left eye face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.wink_right

Detected wink of right eye face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.brows_down

Detected eyebrows down face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.brows_up

Detected eyebrows up face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.teeth

Detected gnash teeth face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.eyes_wide

Detected eyes wide open face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.mouth_down

Detected frowning lips face expression (number between 0-255 corresponds to 0-100%)

byte kortexino.eyes_closed

Detected both eyes closed face expression (number between 0-255 corresponds to 0-100%)

int kortexino.compase

Compass heading in degrees

int kortexino.gyro_x

Direction of gravity (x component of 3D vector) between -100 and 100 (corresponding to -1 and 1)

int kortexino.gyro_y

Direction of gravity (x component of 3D vector) between -100 and 100 (corresponding to -1 and 1)

int kortexino.gyro_z

Direction of gravity (x component of 3D vector) between -100 and 100 (corresponding to -1 and 1)

Functions:

void kortexino.read_data()

Read data received via Bluetooth into the kortexino object.

The Robot Object:

Functions:

void robot.start()

Initialize the Romeo BLE micro controller.

void robot.move_forward(char motor_speed)

Set the speed of both motors to move the robot forward. char motor_speed: set to a number between 0 and 150 to control motor speed.

void robot.move_backward(char motor_speed)

Set the speed of both motors to move the robot backwards. char motor_speed: set to a number between 0 and 150 to control motor speed.

void robot.turn_right(char motor_speed)

Set the speed of both motors to turn the robot right. char motor_speed: set to a number between 0 and 150 to control motor speed.

void robot.turn_left(char motor_speed)

Set the speed of both motors to turn the robot left. char motor_speed: set to a number between 0 and 150 to control motor speed.

void robot.stop_movement()

Set the speed of both motors to stop their movements.

void robot.set_motor(byte motor, char motor_speed, bool forward)

Set the speed of one motor. byte motor: set to 1 or 2 to control motor 1 or motor 2; char motor_speed: set to a number between 0 and 150 to control motor speed; bool forward: set to true for forward movement, false for backward movement.

You can use combinations of this function to control any movement of the two motors.