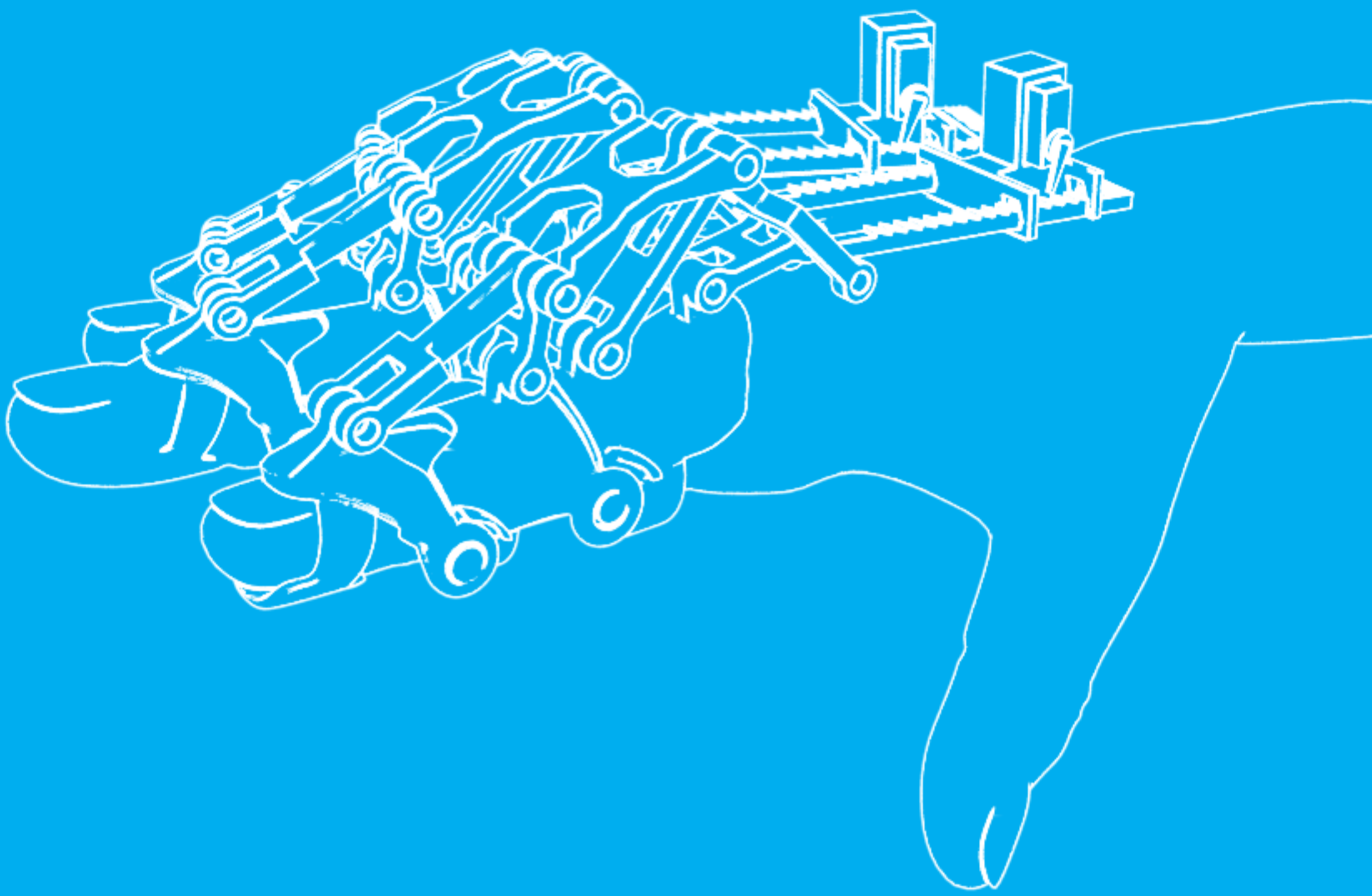
Create your own Sense Glove for Virtual Reality



00

Table of Content

CAUTION

All data and information provided in this guide is for informational purposes only. we make no representations as to accuracy, completeness, currentness, suitability, or validity of any information on this site and will not be liable for any errors, omissions, or delays in this information or any losses, injuries, or damages arising from its use.

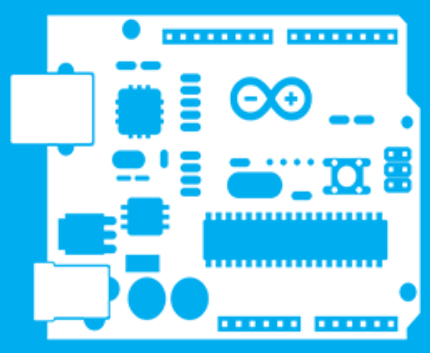
01

Things you need

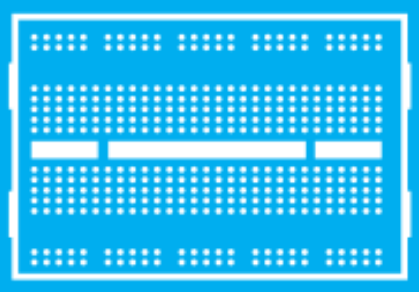
Files ☐ Visit our github page for all the files that you need to start building the sense glove. This includes the 3D model, the Unity scenes and the source code for the Arduino to work.

3D printer ☐ You need access to an 3D printer to print out the skeleton for the glove.

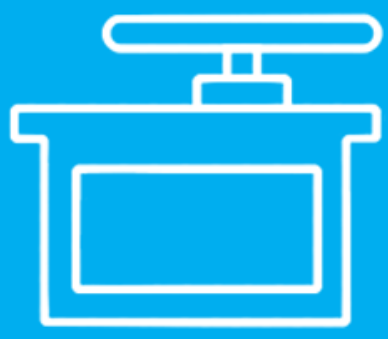
Shopping list ☐ In order to get your glove pieces together and connect it with the game engine you need the following things:



Arduino uno + shield



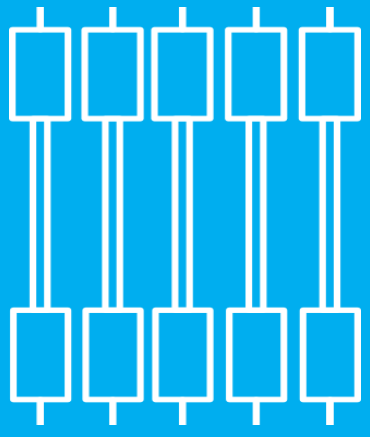
Breadboard



2X Servo motor

30X Nuts & Bolts

size depends on your model scaling



Connection Wires

Leap Motion

Printer Cable

WD40

02

Printing the model

For the printing of the Sense Glove we have used an XYZ Davinci Junior 3D Printer. This is one of the cheapest full assembled printers and doesn’t require any technical knowledge.
All of the parts can be printed within 15x15x15 cm.

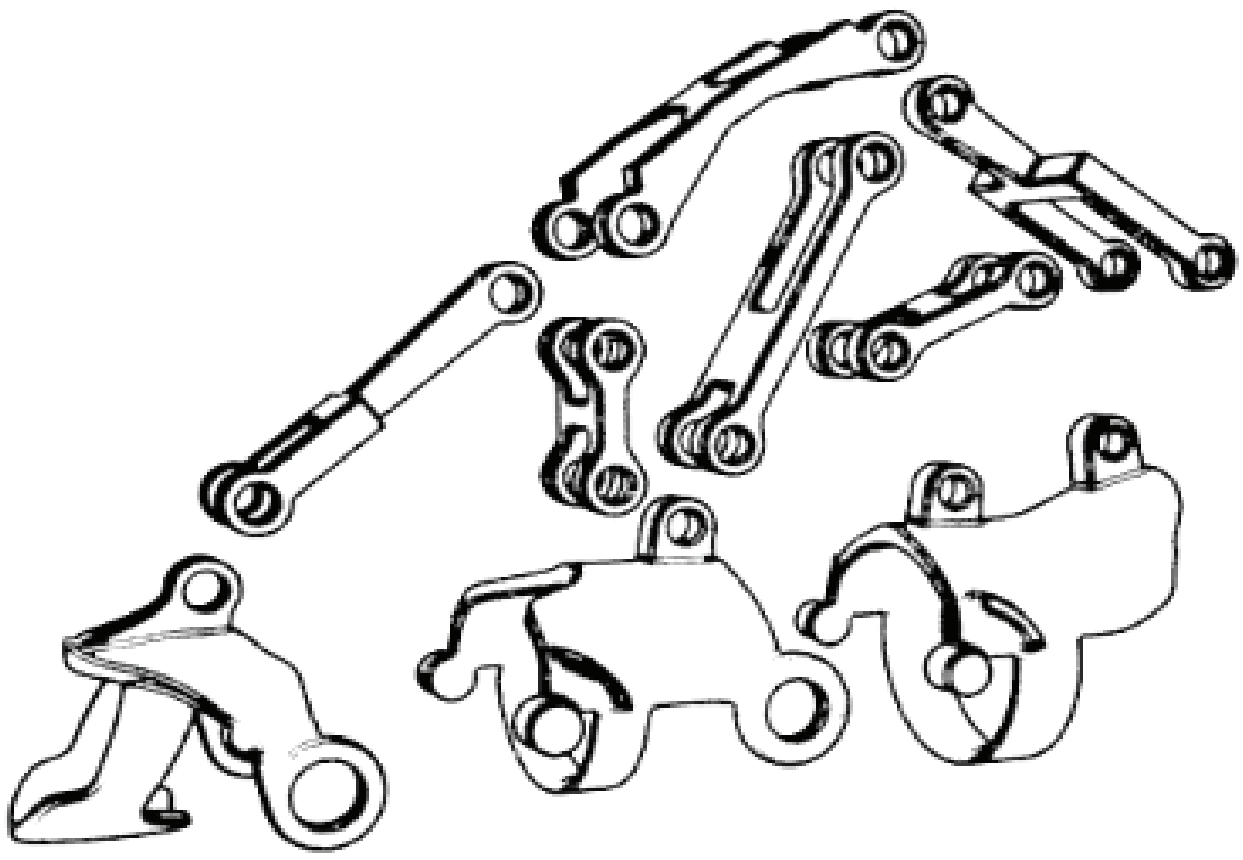
For printing we used 30% - 50% infill depending on the forces the part will endure. Also almost all the parts use supports and rafts for printing so you can be sure the model doesn’t break during the printing process. We use a resolution layer height of 0.3mm/300 microns for maintaining the quality of the models.

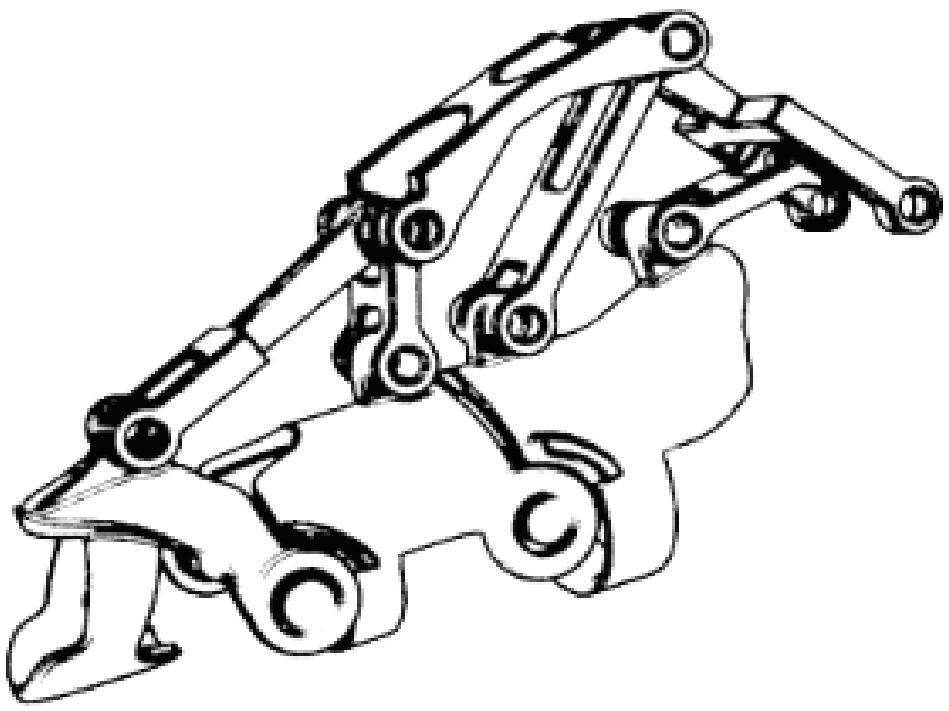
Note: It is important to test-print some of the objects until it is relative to the size of your hand.

03

Assembling

For assembling the Sense Glove you have to make sure all the parts are printed based on the scale of your hands. We used 30mm * 2mm screws with a very thin screw-wire.
This makes sure we can lock it to the parts with bolts and the screw-wire won't damage the hinges. In the image below you can see how the parts should be constructed together.

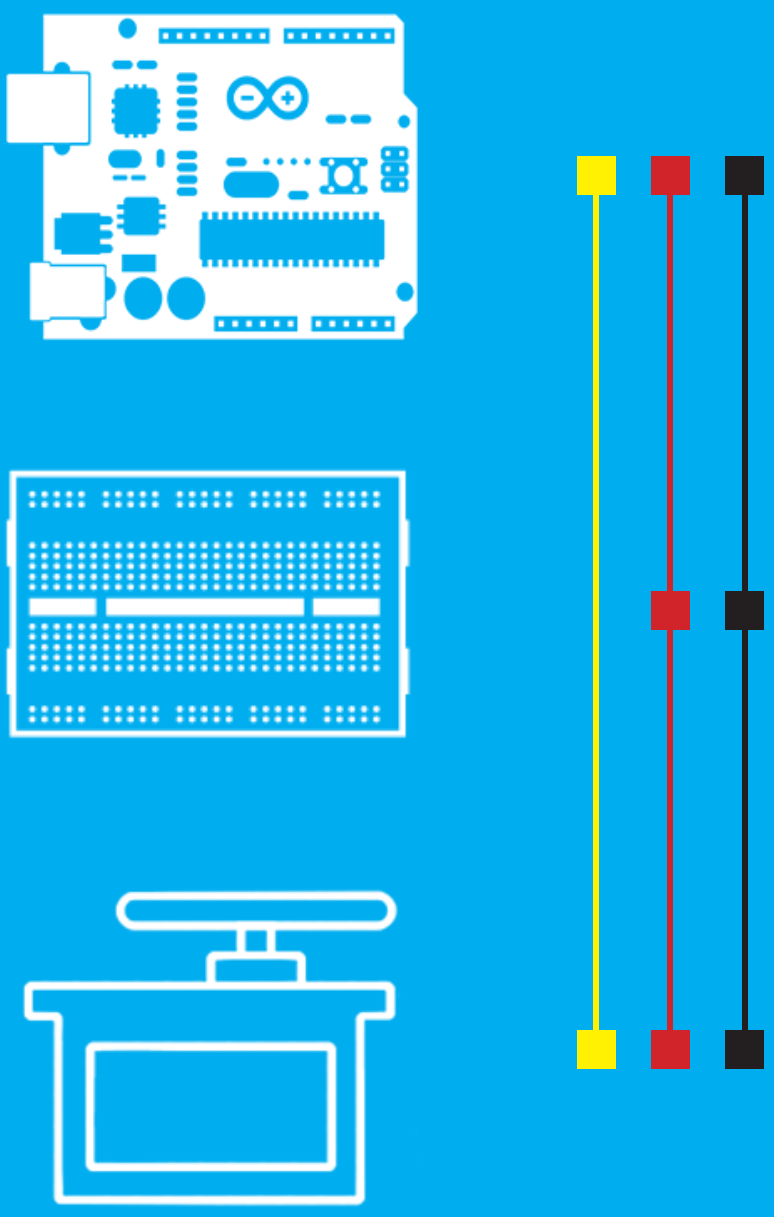




04

Connecting Arduino

We start by placing the shield on the Arduino, the connectors should guide you for the rotation it should be placed. After that we connect the 2 Data wires to the Analog Outputs 9 and 10. The other side of these wires will be connected to the female orange wire inputs of the servo. Then we put an 5 Volt wire into the 5V Output of the Arduino. This wire will be placed on the other side in the breadboard. The location of the 5V wire in the breadboard doesn't matter, but you have to put the other 2 5V wires in the same horizontal row as the one which comes from the Arduino. After you have connected those wires you can put the other sides in the Servo's power input wires (also red wires).
Do the same process for the Ground wires (black) except place the wire which goes to the Arduino in the GND output. And place the Ground wires into the brown female inputs of the servos.



Below is the script we use for moving one of the servos using Unity it should be easy to expand for multiple servos. This script receives through the USB-COM port different strings containing one letter, the “f” (free) or the “s” (stuck). This means that Unity sends data when the hand collides with an object using the Leap Motion Technology. If the Arduino receives the “s” the servo closes and locks the fingers, but i when Unity sends an “f” the servo opens again. Also to debug the system the LED-light turns on when the Arduino receives an “s”.

Arduino Sketch

```
#include <Servo.h>

Servo myservo;
int data;
int pos = 0;
char myCol[20];

void setup() {
  Serial.begin(9600);
  pinMode (13,  OUTPUT);
  myservo.attach(9);
}

void loop() {

  int lf = 10;
  Serial.readBytesUntil(lf, myCol, 1);

  if (strcmp(myCol, “s”) == 0) {
    digitalWrite(13, HIGH);
    for (pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
    { // in steps of 1 degree
      myservo.write(pos);              // tell servo to go to position in variable ‘pos’
      delay(1);                       // waits 15ms for the servo to reach the position
    }
  }

  if (strcmp(myCol, “f”) == 0) {
    digitalWrite(13, LOW);
    for (pos = 180; pos >= 1; pos -= 1) // goes from 180 degrees to 0 degrees
    {
      myservo.write(pos);              // tell servo to go to position in variable ‘pos’
      delay(15);                       // waits 15ms for the servo to reach the position
    }
  }
}
```

Compile and Upload the scripts using Arduino Sketch to the Arduino and it should work.

05

Connecting to Unity

We start by connecting the Leap Motion Controller to Unity. We have to download Leap Motion Drivers, Leap Motion Orion Beta, Unity 5.4+ and Leap Motion Unity Core Asset package.

We start by installing the Leap Motion Drivers and Unity 5.4. After that we install Leap Motion Unity Beta. We open Unity and import the Leap Motion Unity Core Asset Package.
In the scene folder we can open scenes and test our Leap Motion Controller.

After that works we can start connecting the Arduino to Unity.
On the next page you can find a script and breakdown of how we connected the Arduino to Unity.

This script sends strings to the Arduino when the spacebar is pressed, but this can easy be replaced by an OnCollisionEnter/OnCollisionExit function.
It is important to declare the USB port using Arduino Sketch > Tools > Port. For this example we use an Macbook so the USB-port is called “/dev/tty.usbmodem1411”, on windows the port is called something like COM5.
The 9600 stands for the Baudrate which means the speed to send data from Unity to the Arduino.

Unity C# script

```
using UnityEngine;
using System.Collections;
using System.IO.Ports;

public class Arduino : MonoBehaviour {

    public bool freedom = true;
    public SerialPort serial = new SerialPort("/dev/tty.usbmodem1411", 9600);

    // Use this for initialization
    void Start () {
        serial.Open ();
        serial.ReadTimeout = 10;
    }

    // Update is called once per frame
    void Update () {

        if (serial.IsOpen) {

            if (Input.GetKeyUp (KeyCode.Space)) {
                freedom = true;
                serial.Write ("f");
                Debug.Log ("free");
            }

            if (Input.GetKeyDown (KeyCode.Space)) {
                freedom = false;
                serial.Write ("s");
                Debug.Log ("stuck");
            }

        }   else if(!serial.IsOpen) {
            Debug.Log("serial isnt open");
        }
    }
}
```