SEM

Figure 1, picture of SEM (and/or CAD drawing?)

Materials

Table 1, Bill of materials for the SEM.

Part	Count, vol. or length	Manufacturer	Manufacturer serial # or *.stl file	Approximate cost, EUR	Part
Arduino nano	1				
Raspberry Pi 4b	1				
Servo motor	2				
5V power supply	2				
wire	500 mm				
Breadboard	1	Bud industries	BB-32650-B	3.30	
Transparent film A4 pieces	2	Staedtler	10DT6F		
Chloroform*	1 ml				
300 mm X 50 mm rectangles of 0.5 mm thick aluminium sheet	2				door
3-d printed parts	1		scale1.stl	7.6 all parts	A
	1		scale2.stl		B
	1		scale3.stl		C
	2		scale4.stl		D
	4		bb_base.stl		E
	4		bb_base_nut.stl		F
	1		bb1_send.stl		G
	1		bb1_receive.stl		H
	1		bb2_send.stl		I
	1		bb1_receive.stl		J
	2		door_base.stl		K
	2		door_base_nut.stl		L
	2		servo_clamp.stl		M
	2		servo_bolt.stl		N
	2		servo_bolt_cap.stl		O
3mm Acrylic sheet parts	4		door_guide_back.stl		P
	4		door_guide_spacer.stl		Q

	4		door_guide_front.stl		R
	4		sem_support.stl		S
	4		prevent_escape.stl		T
	1		floor.stl		U
	2		door_lever.stl		V
20 mm square profile aluminum rail, 500 mm length	6				
20 mm square profile aluminum rail, 300 mm length	4				
M6 cap screws, 30 mm length	6	McMaster-Carr	90128A266	9.2	
M6 grub screws, 30 mm length	8				
M6 Drop-In T-Nut	8	thorlabs	XE25T1/M		
M3 cap screws, 30 mm length	4	McMaster-Carr	91290A171	7.2	
M3 cap screws, 10 mm length	2	McMaster-Carr	91274A105	6.4	
M3 nuts	2				

Approx. total cost:

Useful tools for assembly are an allen key set, a small Phillips head screw driver, wire cutters, a soldering iron, 1 ml syringe for the chloroform, personal protective equipment and epoxy glue.

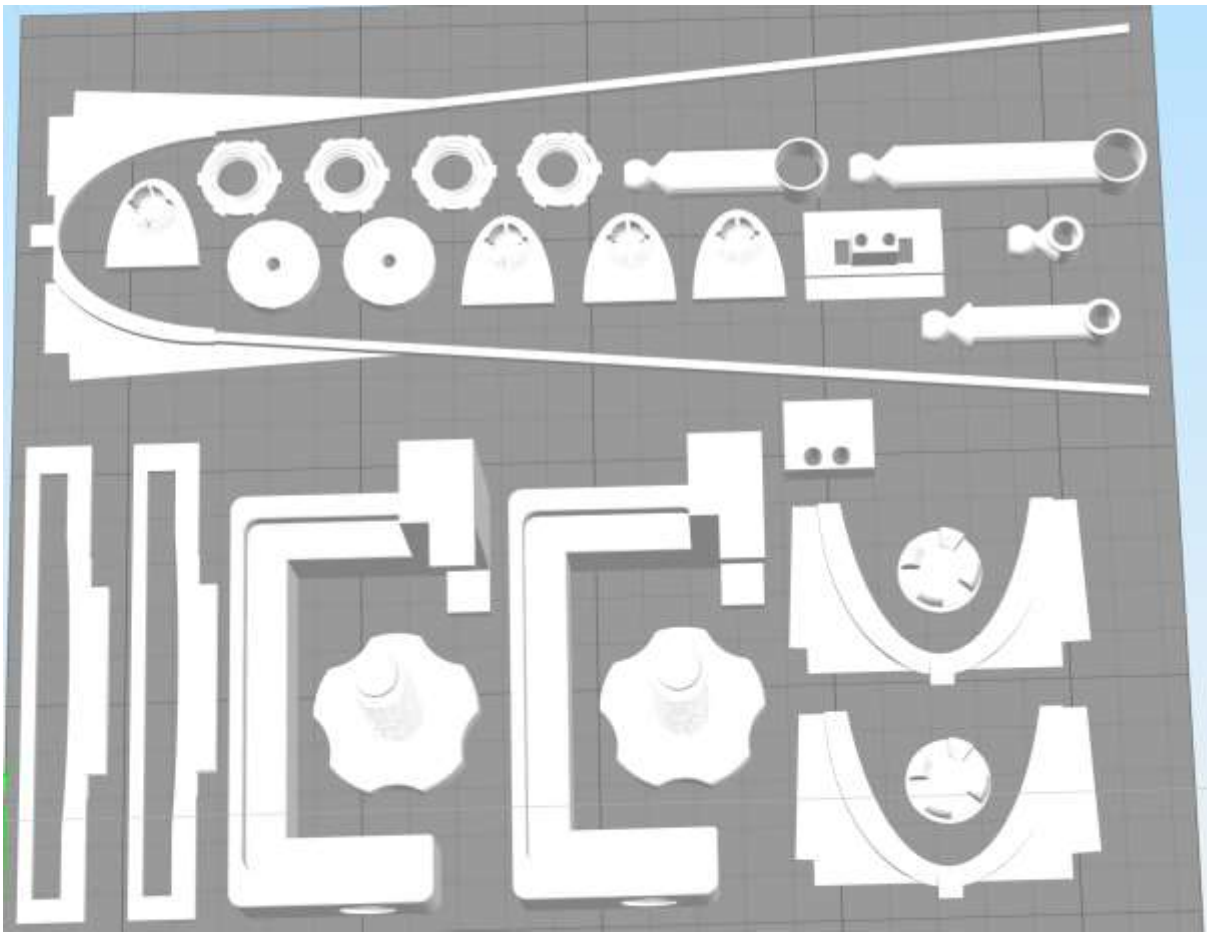


Figure 2, 3D printed parts.

Mechanical assembly

More assembly detail needed!

The single entry module (SEM, Figure 1) consists of a 200 mm long U-shaped corridor, constructed from a sheet of transparent, firm, light-weight plastic (made from two overhead projector transparency sheets) which is secured into a U-shape via 3D-printed anchors (PLA). The U-shaped corridor is screwed onto a 100 g load cell (sparkfun TAL221 SEN-14727) which is secured to . Laser cut 3 mm acrylic sheet is used to construct edge guides for the two doors (Figure 2) by chemical welding using chloroform (using appropriate PPE). Additional pieces of acrylic sheet are attached to the door edge guides to support the edges of the U-shaped corridor in case an animal pushes against it. The SEM can be closed off on both ends via a vertical sliding door constructed from a XX x XX x XX aluminum sheet which can be lowered into the floor of the maze via a servo motor attached to a beam below the maze. An RFID-tag reader (sparkfun XXX) is placed below the U-shaped corridor, near door 2. Two beam break devices (BB), constructed from a xx nm laser and photoresistor circuit mounted on 3D printed ball joints (Figure 3) are used to ensure safe operation of the doors. One (BB1) is directed 10 mm above door 1 when it is open. It is used to prevent the door 1 closing with an animal on top. This beam is unbroken when the first door is open and no animal is sitting in the opening. Another (BB2) is placed on opposing sides of the corridor so the beam is directed through the corridor at a safe distance (>100 mm) from door 2. A break in this beam signals the exit through the module back to the home cage.

Operation of the module is as follows and corresponds to the MODEs in Raspberry Pi code SEM_only.py:

1) SEM is open for entry from home cage
2) An animal is detected with the RFID reader. If the weight is in range of one animal and no animal is detected on top of door 1, door 1 is closed, trapping one animal in the SEM. Otherwise wait for next RFID detection.
3) The weight is determined by averaging 50 consecutive reads. If the weight is in range of one animal, door 2 is opened and a minimum time out (minimum_entry_time) is waited until exit detection ensues.
4) When the animal is returning to the home cage through the SEM, BB2 is broken, and consequently door 2 closes, door 1 opens and the SEM is open for the next entry (MODE 1).

Assembly steps:

1) Build a support frame for the floor plate using 500 mm long 20 mm square profile aluminum rail according to Figure 3. A minimum height of 500 mm from the floor is recommended as the doors need to operate some distance below the floor.

Figure 3. CAD of frame

2) Assemble the doors:

Using the drop-in T-nuts and M6 grub screws hang a 300 mm aluminum rail downward from the support frame, and attach another 300 mm aluminum rail perpendicular to that (Figure 4). Attach a servo motor on top of the perpendicular rail using 3D printed clamp (parts M-O).

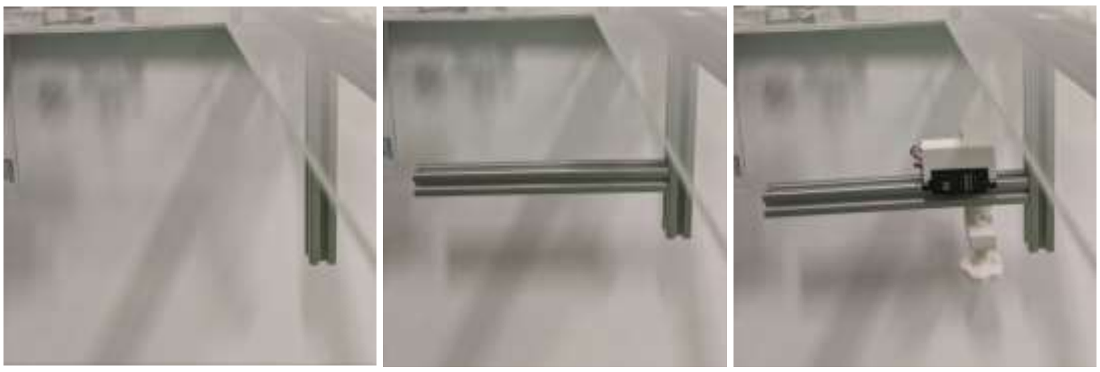


Figure 4, servo motor installation.

Prepare the moving parts (Figure 5): Attach a lever made of acrylic sheet (part V) to a servo hub (provided with the servo) using a 10 mm M3 screw. Once tightened, working in a well ventilated space/chemical hood add 1-2 drops of chloroform on the joint to activate the surfaces. After a few minutes add 2-3 drops of superglue to the joint. After the glue has set, unscrew the M3 screw. Slot a 300 mm X 50 mm rectangle of 0.5 mm thick aluminum sheet (part door) into a 3D printed base (part K) and add 2-3 drops of superglue to the joint. A small amount of epoxy glue can be used to bond these glued joints as they will carry variable loads. Attach a 30 mm M3 cap screw and nut to the other end of the lever.

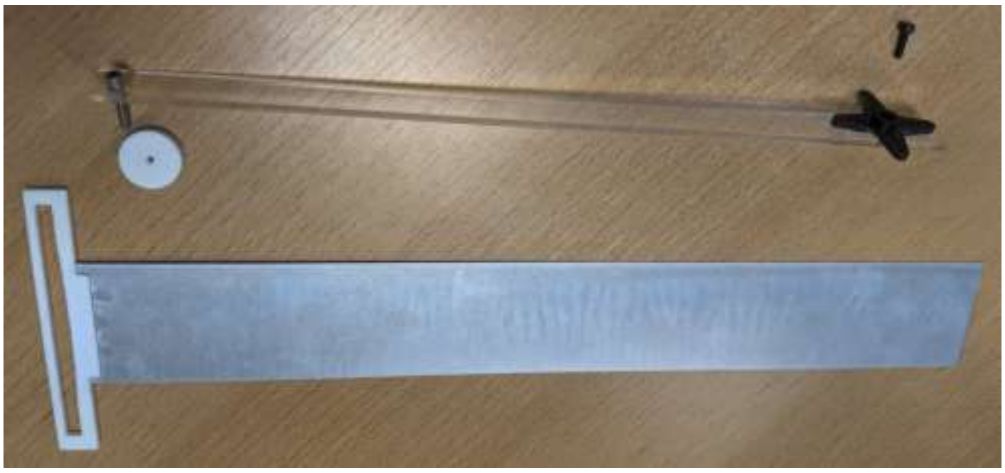


Figure 5, acrylic door lever along with attachment pieces (above) and aluminum door (below).

Attach the lever to the servo such that its movement arc cannot collide with the floor. Slot the door through its aperture in the floor and attach it to the lever as shown in Figure 6 using the 3D printed nut (part L) to cap the 30 mm screw but do not tighten it against the door base.

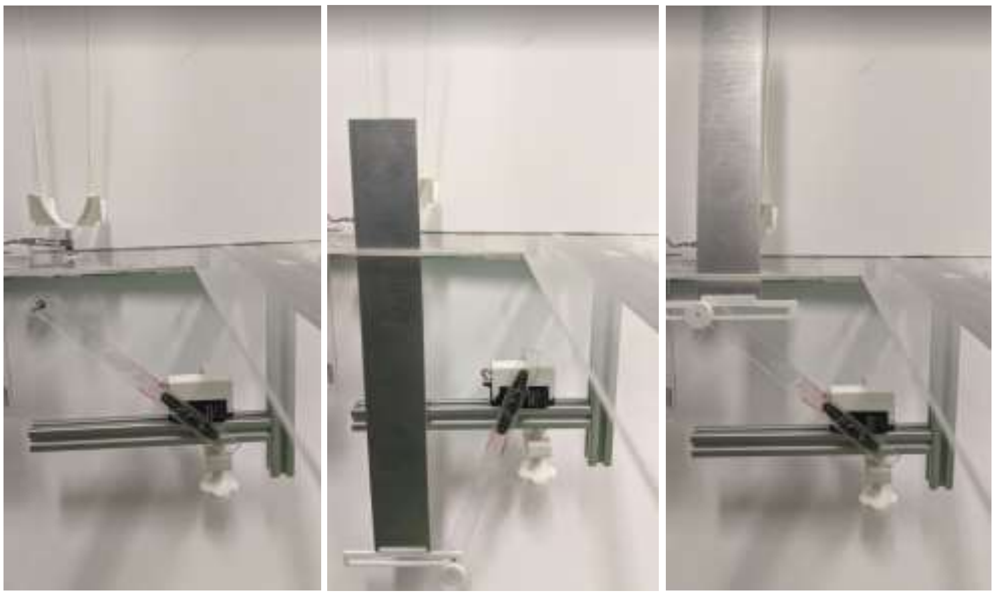


Figure 6, Door installation.

Then door guides

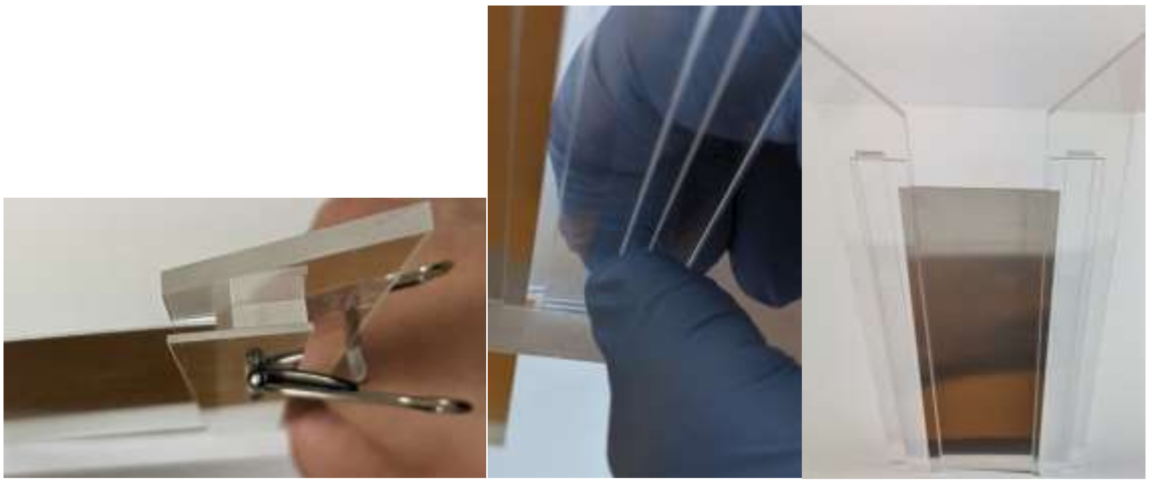
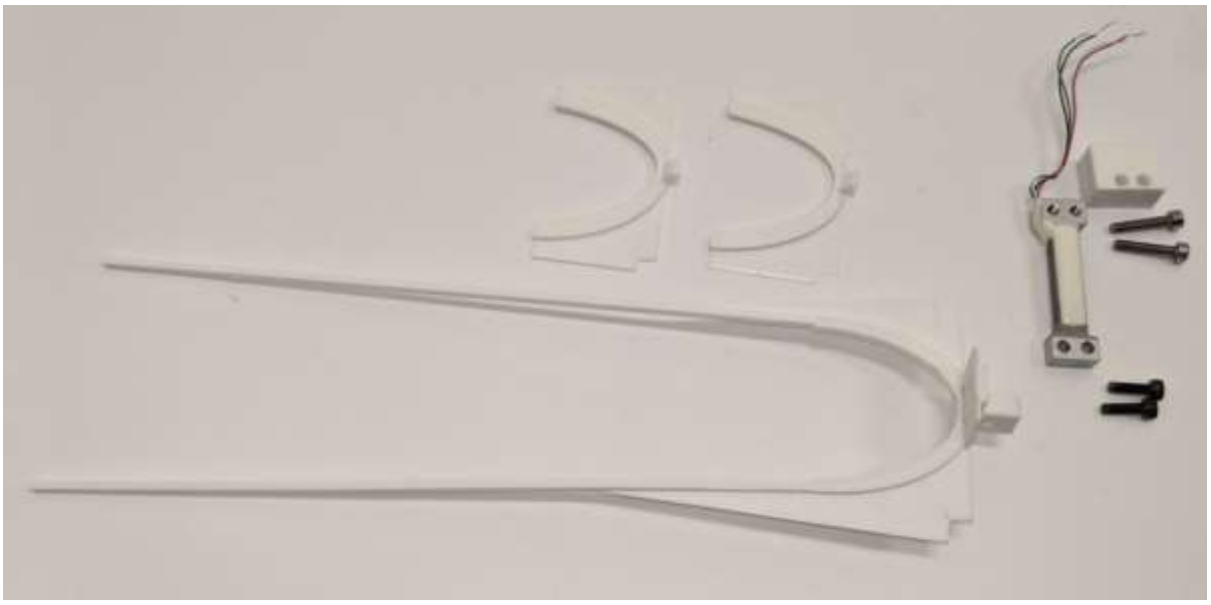
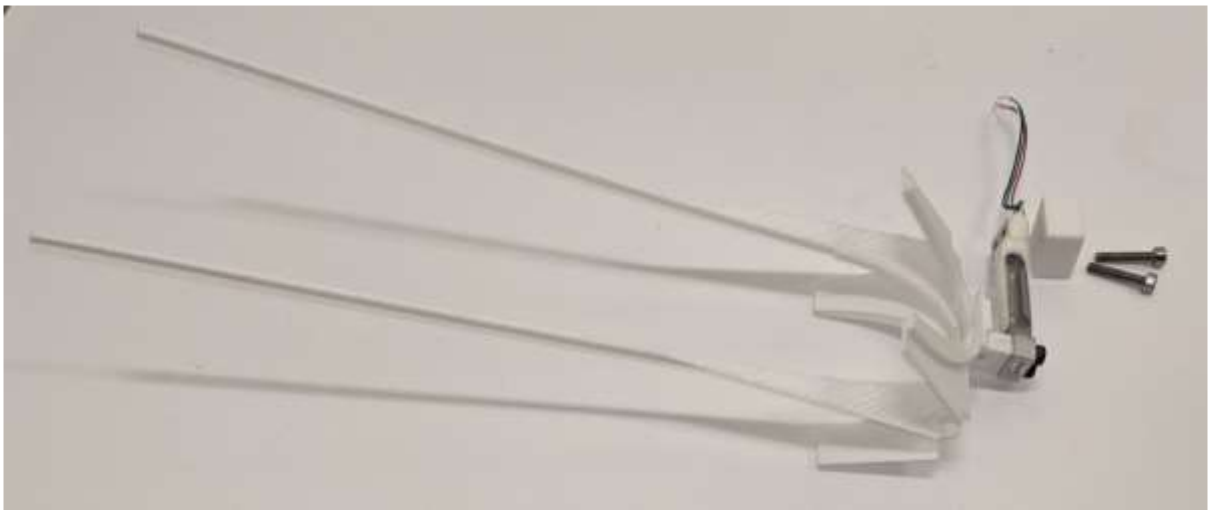


Figure 7

Guide frames for the door are installed using acrylic sheet pieces

Then the U form





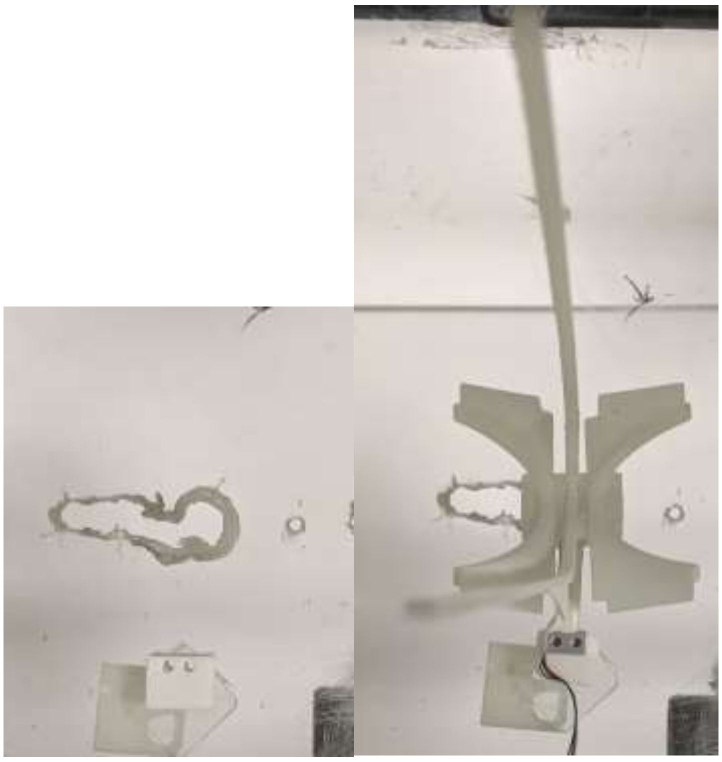
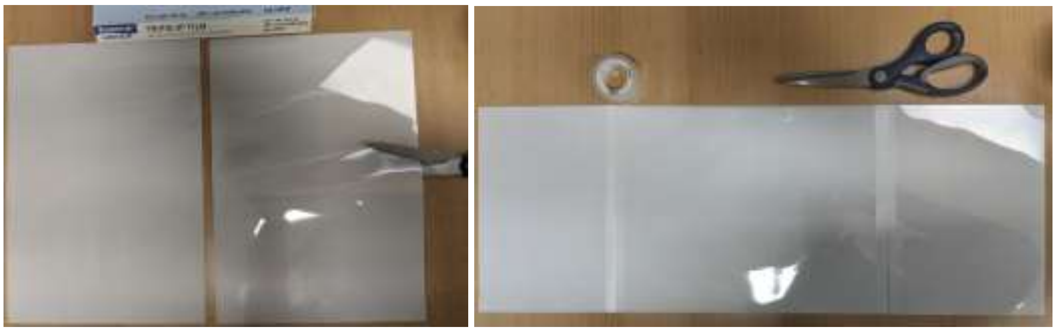
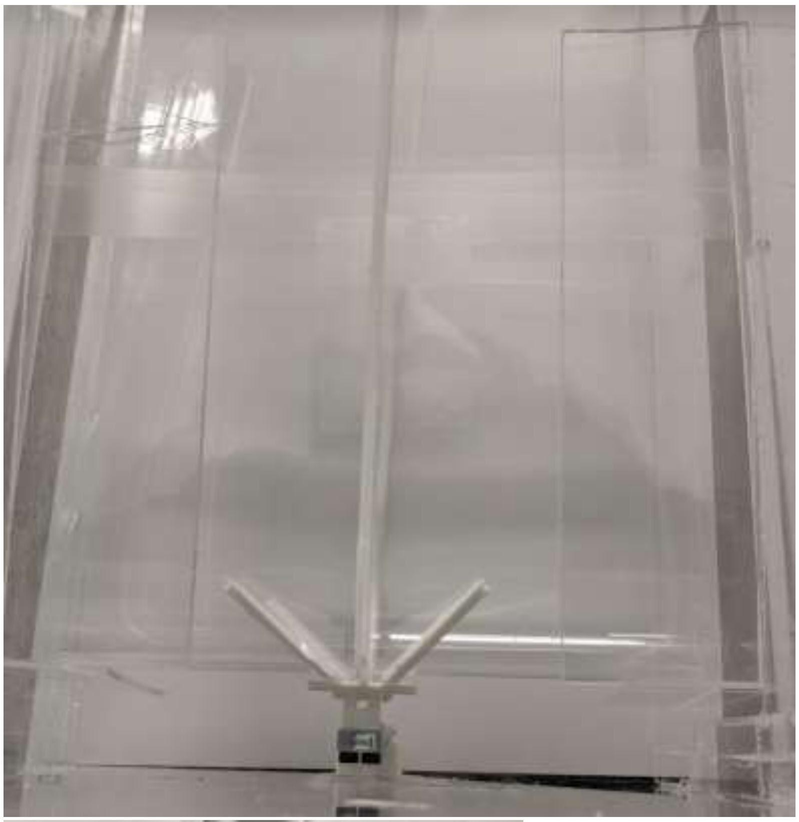


Figure 8, the scale backbone.

A 597 x 200 mm rectangle of transparency film is made by taking two A4 sheets, cutting one in half and taping to the ends of the intact one, followed by cutting 10 mm off the long edge (Figure 7).

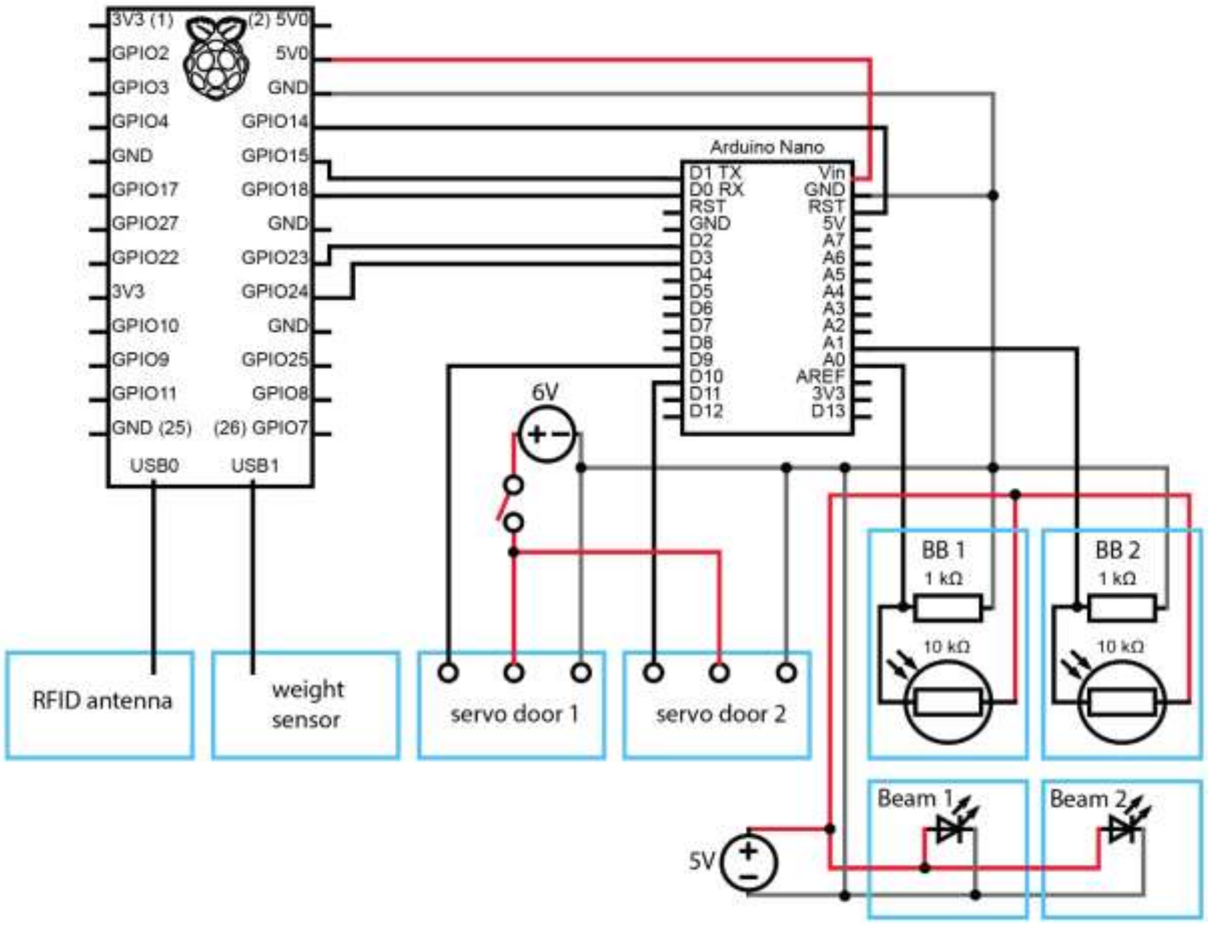




Then U form supports

And finally beam breaks.

Electronics



Code

Arduino

An Arduino board is used to detect beam breaks, reporting them to the Raspberry Pi, and open/close doors when commanded by the Raspberry Pi. The following code (MaheshKarnani, 2022a) is flashed to an Arduino Nano (also Uno and Mega boards were tested):

```
#include<Servo.h>
Servo servo1;     // Servo 1 = door1
#include<Servo.h>
Servo servo2;     // Servo 2 = door2

//Constants
const int slowness = 150;   //slowness factor, us wait between 1deg movements, change to set door
speed
```

```
//beam breaks
const int pResistor1 = A0;//BB1 safety
const int pResistor2 = A1;//BB2 return

//servo control
const int servoPin1 = 9;// door servos 1-2
const int servoPin2 = 10;

//outputs to Pi
const int ard_pi_BB1 = 0;//reports BB1low to Pi
const int ard_pi_BB2 = 1;//reports BB2low to Pi

//door angles: determine empirically
const int CLOSE_DOOR1 = 150;//Angle of 150 degrees -> door is closed
const int OPEN_DOOR1 = 47;//Angle of 47 degrees -> door is opened
const int CLOSE_DOOR2 = 164;//Angle of 155 degrees -> door is closed
const int OPEN_DOOR2 = 30;//

//motor inputs from Pi
const int pi_ard_door1 = 2;//open door1
const int pi_ard_door2 = 3;//open door2

//Variables
int photo_value1;//Store value from photoresistor (0-1023)
int INIT_READ1;//Store initial value from photoresistor
int photo_value2;//Store value from photoresistor (0-1023)
int INIT_READ2;
int pos1_current = CLOSE_DOOR1; //initial position variables for servos
int pos1_target = CLOSE_DOOR1;
int pos2_current = CLOSE_DOOR2; //initial position variables for servos
int pos2_target = CLOSE_DOOR2;
long start = millis();


void setup()
{
   //Serial.begin(9600);//setup serial
  pinMode(pResistor1, INPUT);//Set photoResistor - A0 pin as an input
  pinMode(pResistor2, INPUT);
  pinMode(ard_pi_BB1, OUTPUT);//output reports to Pi
  pinMode(ard_pi_BB2, OUTPUT);
  pinMode(pi_ard_door1, INPUT);//input commands from Pi
  pinMode(pi_ard_door2, INPUT);

  servo1.attach(servoPin1);
  servo1.write(OPEN_DOOR1);
  delay(1500);
  INIT_READ1 = analogRead(pResistor1);//calibrate top of door detector when door1 open
  delay(200);
  servo1.write(CLOSE_DOOR1);
  delay(100);
  servo2.attach(servoPin2);
  servo2.write(OPEN_DOOR2);
  delay(100);
  servo2.write(CLOSE_DOOR2);
  delay(100);

  digitalWrite(ard_pi_BB1, LOW);//communication to Pi
  digitalWrite(ard_pi_BB2, LOW);

  INIT_READ2 = analogRead(pResistor2);// calibrate exit beam-break

//  Serial.print("INIT_READ1 ");  //show beam break values for trouble shooting if serial monitor is on
//  Serial.println(INIT_READ1);
//  Serial.print("INIT_READ2 ");
//  Serial.println(INIT_READ2);
}

void loop()
{
   photo_value1 = analogRead(pResistor1); //read beam breaks
   photo_value2 = analogRead(pResistor2);

   // Serial.print("photo_value1 ");
   // Serial.println(photo_value1);
   // Serial.print("photo_value2 ");
   // Serial.println(photo_value2);

   //output BB detectors to Pi
   if (photo_value1<INIT_READ1*0.8) //BB1 safety - inverted here
   {
      digitalWrite(ard_pi_BB1, LOW);
   }
   else
   {
      digitalWrite(ard_pi_BB1, HIGH);
   }

   if (photo_value2<INIT_READ2*0.7) //BB2 exit
   {
      digitalWrite(ard_pi_BB2, HIGH);
   }
   else
   {
      digitalWrite(ard_pi_BB2, LOW);
   }
```

```
//servo movement loops
if (pos1_current<pos1_target) //SERVO 1
{
   pos1_current=pos1_current+1;
   servo1.write(pos1_current);
   delayMicroseconds(slowness);
}
if (pos1_current>pos1_target)
{
   pos1_current=pos1_current-1;
   servo1.write(pos1_current);
   delayMicroseconds(slowness);
}

if (pos2_current<pos2_target) //SERVO 2
{
   pos2_current=pos2_current+1;
   servo2.write(pos2_current);
   delayMicroseconds(slowness);
}
if (pos2_current>pos2_target)
{
   pos2_current=pos2_current-1;
   servo2.write(pos2_current);
   delayMicroseconds(slowness);
}

//target commands
//SERVO 1
if ((digitalRead(pi_ard_door1) == HIGH))
{
   pos1_target=OPEN_DOOR1;
}
else
{
   pos1_target=CLOSE_DOOR1;
}
//SERVO 2
if ((digitalRead(pi_ard_door2) == HIGH))
{
   pos2_target=OPEN_DOOR2;
}
else
{
   pos2_target=CLOSE_DOOR2;
}
```

}//void loop end

Raspberry Pi

A Raspberry Pi 4b is set up to log relevant events in the SEM and control the entry logic. Python 3.7.3 and the python libraries specified in the requirements.txt file here (MaheshKarnani, 2022b) need to be installed. The helper script SEM_functions.py should be in the same folder as the main script SEM_only.py. The PiGPIO library is launched as a daemon by typing 'sudo pigpiod' in the command prompt. Then the following script is run (we use Thonny Python IDE):

```python
# Single entry module from Switch_maze
from SEM_functions import *
"""
Execution loop for single entry module demo.
Opens, detects one animal in module and reads its RFID, if animal hasn't
exited in 100s closes safely,
weighs animal, passes to other side, waits minimum_entry_time, then allows
return.
Changes typically not recommended here.
This is an example.
"""
while True:

    # reset
    if MODE == 1:
        pi.write(pi_ard_door1, 1)  # open SEM
        print("\nSEM open\n")
        for x in range(np.size(animal_list)):
            animaltag = animal_list[x]
            tick = pi.get_current_tick()
            save.append_event("*", "", "entry_available", animaltag, tick)
        MODE=2

    # wait for entry
    if MODE == 2:
        print("Scanning RFID tag in scale")
        print(datetime.datetime.now())

        while True:
            animaltag = RFID_readtag("RFID1")
            if animaltag:
                w = read_scale()
                if (
                    w > 10
                    and w < heavy
                    and pi.read(ard_pi_BB1)
                    and int(round(time.time()))
                    - animal_timer[animal_list.index(animaltag)]
                    > nest_timeout
                ):
                    pi.write(pi_ard_door1, 0)  # close door1
                    MODE = 3
                    choice_flag = False
                    entry_flag = False
                    water_flag = True
                    break
            else:
                MODE = 2
                print("*", end=",")
                break

    # correct animal on scale
    if MODE == 3:
        print("\nweighing\n")
        # Weighing for entry
        flag_heavy = acquire_weight(animaltag)
        if flag_heavy:
            # Append data
            tick = pi.get_current_tick()
            save.append_event(
                "+", "", "ENTRY DENIED MULTIPLE ANIMALS", animaltag, tick
            )
            MODE = 1
        if not flag_heavy:
            pi.write(pi_ard_door2, 1)  # open door 2
            tick = pi.get_current_tick()
            save.append_event(cycles_str, "", "entry", animaltag, tick)
            print("\nPASS\n")
            print(animaltag)
            print(datetime.datetime.now())
            time.sleep(minimum_entry_time)
            MODE = 4

    # wait for exit
    if MODE == 4:
        print("waiting for exit")
        print(datetime.datetime.now())
        if not pi.read(ard_pi_BB2)
            pi.write(pi_ard_door2, 0)  # close door 2
            animal_timer[animal_list.index(animaltag)] =
int(round(time.time()))
            MODE = 1
```

Adjustment

Beam targeting

Beam break 1 is used for safely closing door 1: Beam 1 is targeted through door 1 such that when the door is open the beam hits the photoresistor. When the door is closed or an animal is on top of the open door, the beam is broken. Beam break 2 is used for detecting an exiting animal at a safe distance from door 2 (>100 mm): Beam 2 is targeted through the sides of the single entry module such that an animal in the middle of the module will break it.

Weight sensor parameter selection and calibration

The weight sensor is set up and calibrated according to the manufacturer's instructions https://learn.sparkfun.com/tutorials/openscale-applications-and-hookup-guide?_ga=2.14390071.1017329722.1674997605-1611569064.1667393722. Briefly, a serial connection to the OpenScale is launched in Arduino IDE and the control menu is accessed by pressing 'x'. Baudrate is set to 19200, report rate 120 ms, units kg, decimals 5, average amount 1, serial trigger 'off'. The scale is then tared to zero and calibrated with a 25 g weight. It is good to repeat the calibration daily in the beginning as the load cell can creep somewhat after installation.

Door angles and speed

The angles that the servo needs to achieve to close and open the door will likely vary based on the exact distances used in each build, and are set by trial and error on lines XXX of the Arduino code. It is best to start with open and closed values near 90 and change them in opposite directions incrementally. In case the door lever collides with something, turn the power to the servos off rapidly from their power switch, go back to the code and bring the degree value closer to 90.

The speed of door movements is set by the slowness constant on line XXX of the Arduino code. This value corresponds to the time, in microseconds, it takes to move the servo by 1 degree. We have found 150 ideal.