Guidance for using automated multiple T maze designed to test cognitive abilities of common shrews (*Sorex araneus*)

Owner: Max Planck Institute for Animal Behavior

Author: Lara Vrbanec

Date: 19.05.2020

Contents

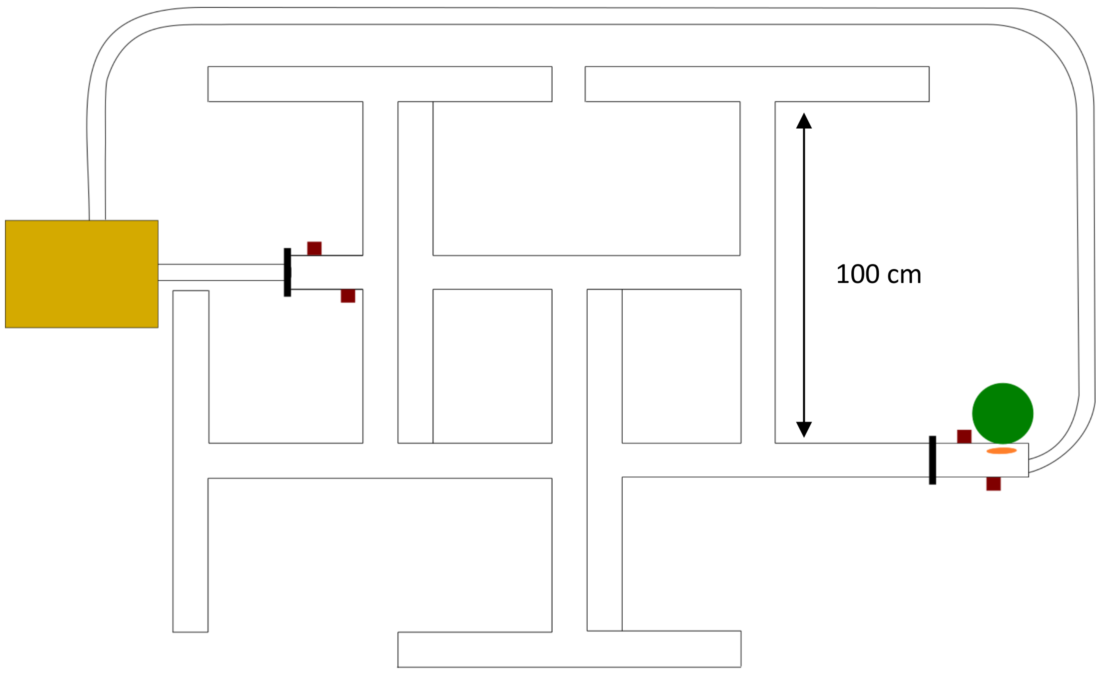
Objective 2
Multiple T maze design 2
Automatic doors design 3
Automatic food dispenser design 4
IR movement sensors design 4
The logic behind the code 5
Electronics components and Supplies 6
Schematics 7
Code 8
Arduino 8
Raspberry pi 12
Procedure for animal testing 16
Analysis 16

Objective

The aim of the multiple T maze is to test common shrews' navigation abilities related to foraging. The animal which makes less mistakes to learn the path which leads to a food reward is considered to be superior in navigation.

Multiple T maze design

Maze arms are built from the opaque grey plastics and covered by the transparent Plexiglas®. The dimension of each arm was 100 x 5 x 5 cm (length, with, height). The maze can be placed into two configurations which are the left - right mirror images of each other. In the first configuration the animal has to perform the sequence of R, L, L, R, R, L turns to escape the maze. In the second configuration has to perform the sequence of L, R, R, L, L, R turns to escape the maze.
As our goal is to test animal's navigation abilities related to foraging, the escaping arm is reinforced with a food reward. In order to simulate natural environment, maze floor is covered with the white sand. The entrance and the goal arm of the maze is connected with the home cage with transparent tubes. The setup includes two automatic doors and one automatic food dispenser. The first automatic door (in the following text referred to as the South door) is placed at the entrance of the maze. The second door (in the following text referred to as the North door) is placed 20 cm in front the food reward. First pair of movement sensors (in the following text referred to as the South sensor) are placed few centimeters behind the South door. The second pair of movement sensors (in the following text referred to as the North sensors) are placed few centimeters being the North door. The automatic feeder is positioned at the end of the goal arm and is used for automatic refill of food award.



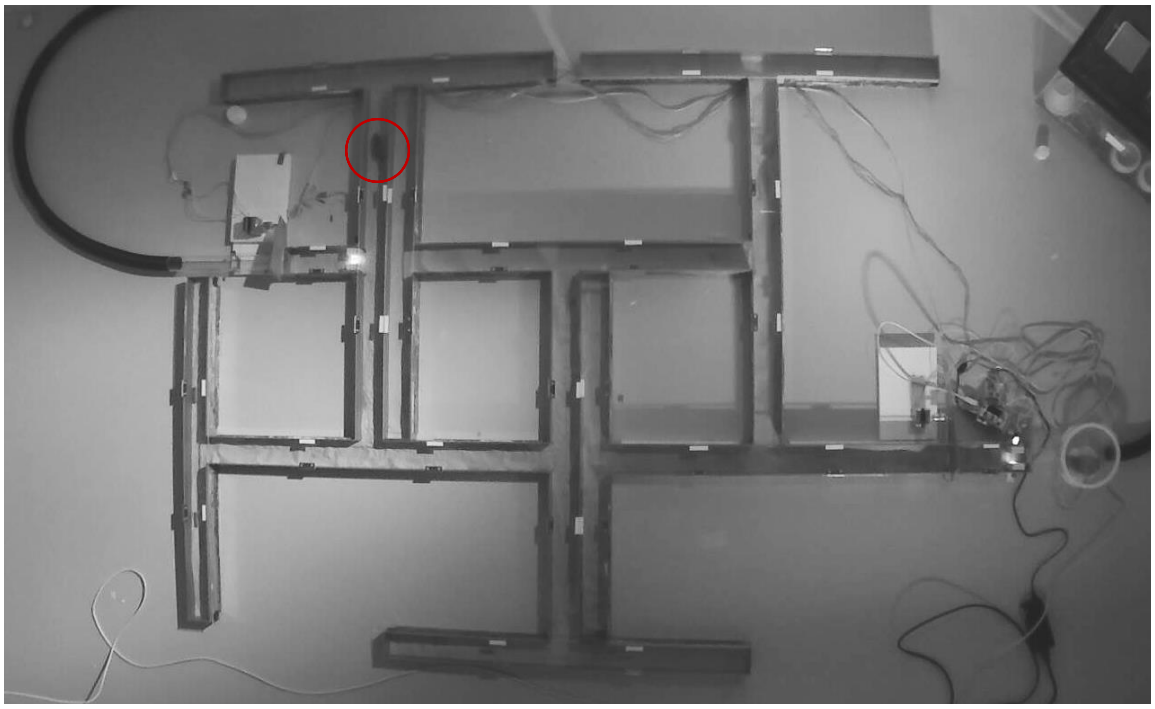
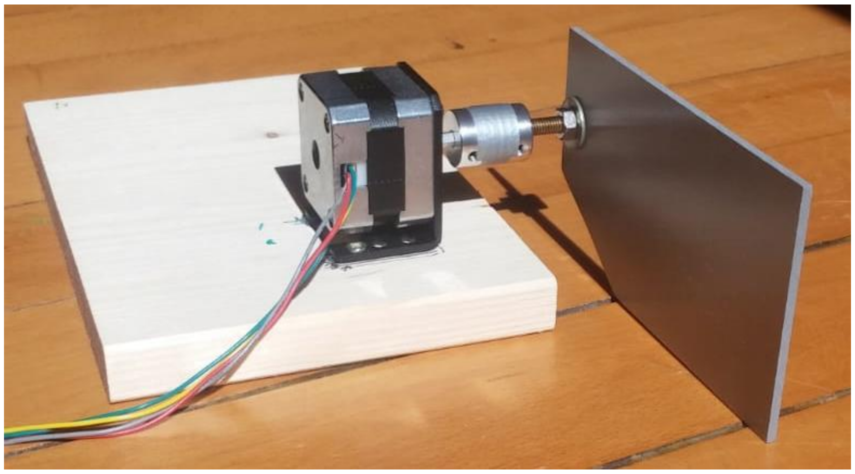


Figure 2. Upper image: Schematic of the multiple T maze in the first configuration (R, L, L, R, R, L). Home cage is shown in yellow, automatic doors in black, movement sensors in red, automatic food dispenser in green, and food reward (mealworm) in orange. Lower image: multiple T maze under low light, recorded with IR camera. The position of the animal is shown with the red circle.

Automatic doors design

A stepper motor arm was prolonged with a screw over motor coupling. Onto prolonged arm, using two mothers, we attached an opaque Plexiglas sheet (2mm thickness, 15 cm length, 7 cm height). Rotation of the motor arm caused rotation of the Plexiglas sheet.



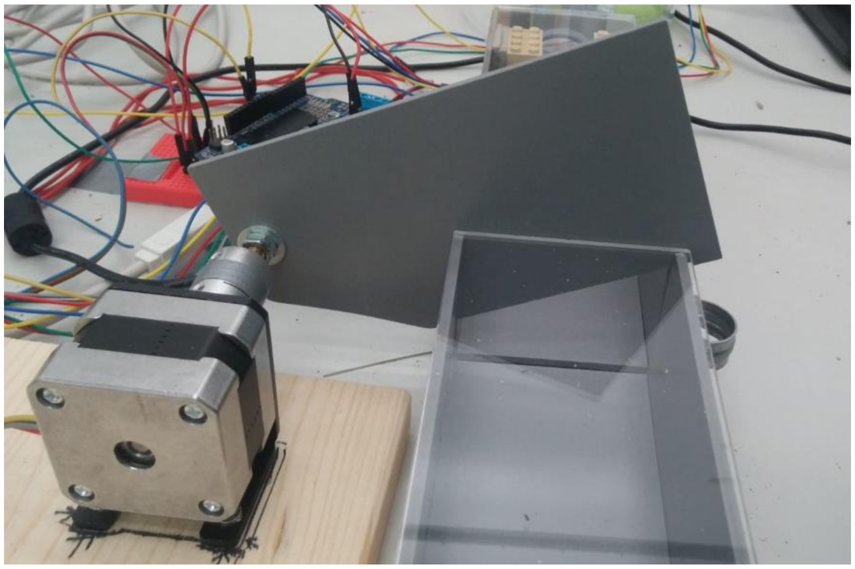


Figure 3. Automatic doors.

Automatic food dispenser design

Automatic food dispenser was built from the manual pill dispenser (example of a pill dispenser https://www.amazon.co.uk/Aidapt-Compact-Weekday-Dispenser-Eligible/dp/B0079N7L8U) by replacing the mechanism for manual rotation with a stepper motor.

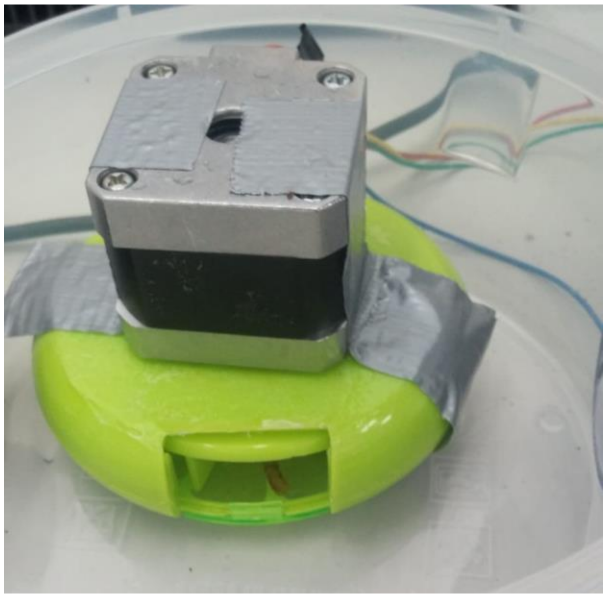


Figure 4. Automatic food dispenser

IR movement sensors design

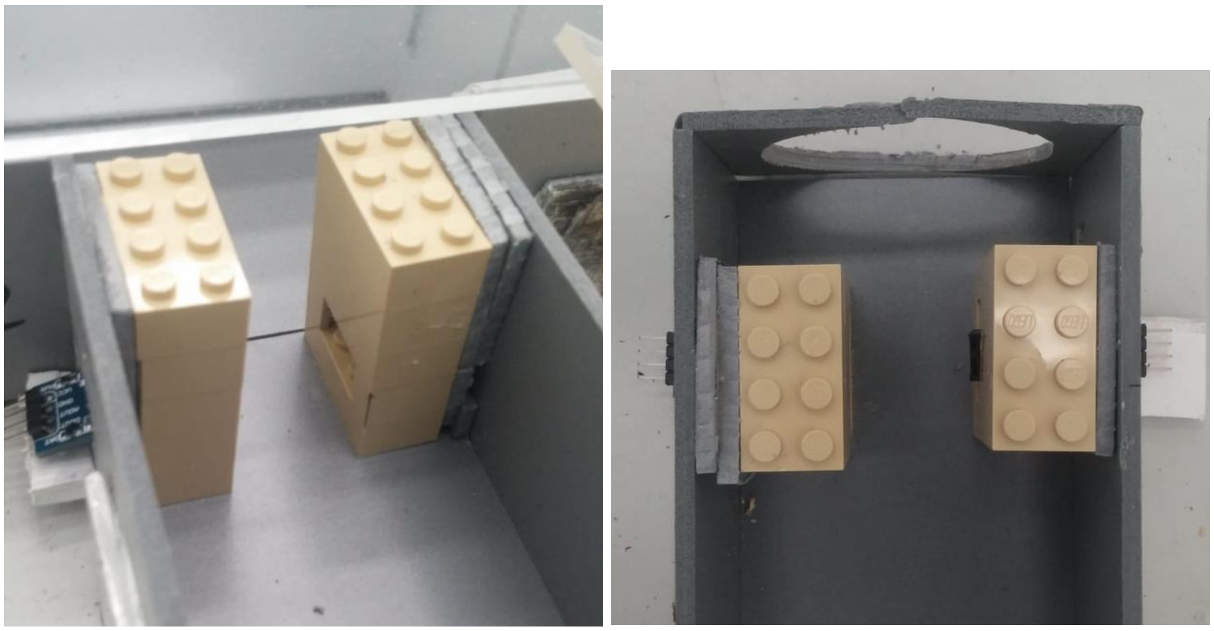


Figure 5. IR movement sensors placed at the beginning and the end of the labyrinth. The structure is L-R symmetrical so the animal cannot use sensor as cue for orientation.

The logic behind the code

When the Arduino Uno reboots, the South doors open, which allows animal to enter the multiple T maze. Animal's entrance in the multiple T maze triggers the South IR movement sensor, which causes automatic feeder to drop the food and the North door to open. When the animal enters the goal arm of the maze and triggers North IR movement sensor, both the North and the South door close. Closing North doors prevents animal from returning to the maze from the opposite direction, and closing the South doors prevents the animal from re-entering the maze immediately after it consumed the food reward. Both doors were closed for 20 minutes. The previous research has shown that common shrews increase their exploration tendencies if when less food deprived. Therefore, we prevented animal from entering the multiple T maze for 20 minutes after it consumed the food reward to avoid testing exploration and to focus on testing navigational abilities during food foraging. The South door automatically opens 20 minutes after they were closed which allows animal to spontaneously enter the multiple T maze from the start arm, which starts the next trial.

In order to track the progress of the experiment without physically being present in the experimental room, following animal's actions trigger the receiving of email notifications.
1) Each time the animal solves the experiment (when it triggers the South IR movement sensor), a message 'Animal has solved the experiment' is received. This email helps the experimenter track the number of trials the animal has conducted.
2) If the animal did not consume food for one hour (if time between consecutive triggering the North IR movement equals 60 minutes or longer), a message 'Animal is starving' is received. This action should warn the experimenter to interfere with the experiment and feed the animal.

Electronics components and Supplies

- 1x Arduino (we used Arduino Uno)

- 1x Raspberry Pi (we used Raspberry Pi 4, 2GB RAM)

- 2x Adafruit Motor Shield V2

second shield needs to be adapted as in instructions on Adafruit https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/stacking-shields

- 2x obstacle avoidance IR sensors (we used Waveshare)

- 3x stepper motor (we used NEMA-17 size - 200 steps/rev, 12V 350mA)
your motor shield needs to be compatible with the power of stepper motors

- Bread board

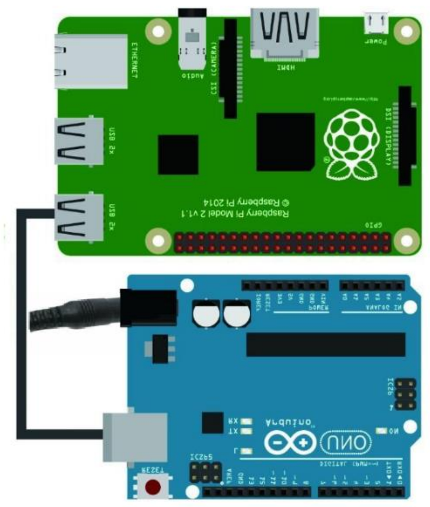
- Jumper cables

- Power source for Arduino (we used 12V/5A) and for Raspberry Pi (we used official SUB charger 5.1V 3A)

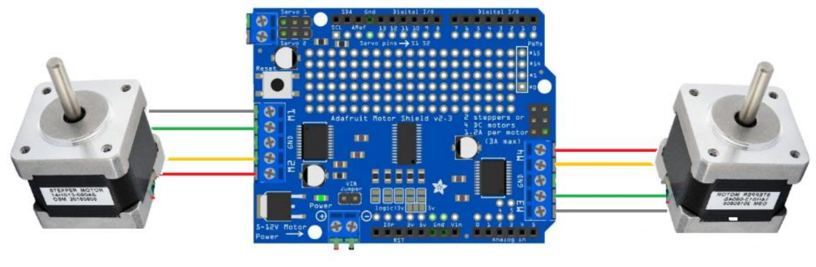
- USB 2.0 cable for connecting Arduino to Raspberry Pi with serial connection

- Ethernet shield and Ethernet connection / WIFI connection for Raspberry Pi

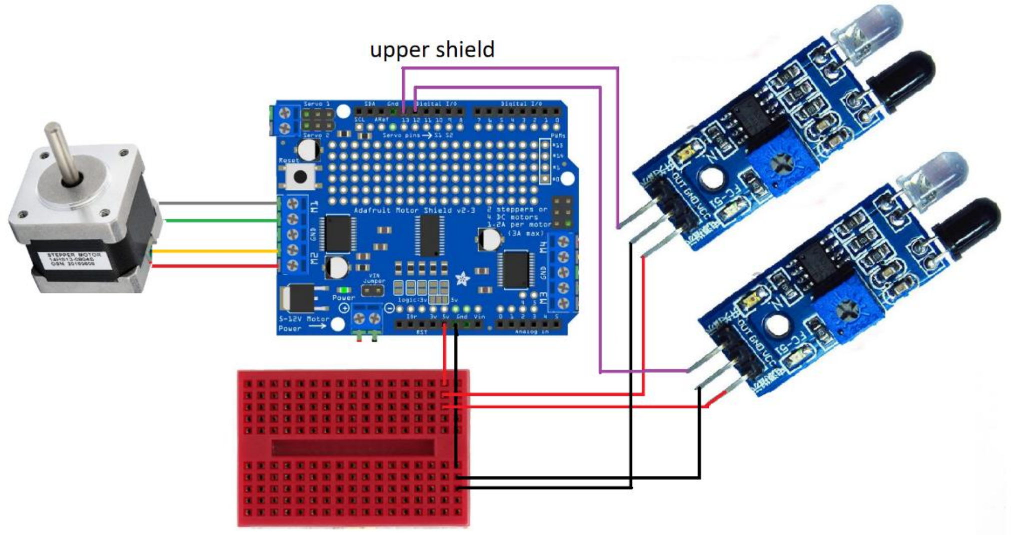
Schematics



lower shield



upper shield



Code

Arduino and Raspberry Pi communicate over serial connection. Arduino is used to move motors when the IR sensors are triggered, and Raspberry Pi is used to send emails to the user.

Arduino

Prior to uploading the code to Arduino change:

```
int starvingT = 30; // define starving time (in ms). If the animal did
not eat for this time, you will receive email notification

int waitingT = 30; // define waiting time (in ms). This is waiting
time between two trials in which the animal cannot enter the maze
(both doors closed)
```

Current values are used for checking the functionality of the setup!

```
// OS Windows 10, Arduino 1.8.10

// Author Lara Vrbanec

// Date 14/04/2020

// code sources Adafruit_motor_shield_v2
https://learn.adafruit.com/adafruit-motor-shield-v2-for-
arduino/stacking-shields

// sensors Waveshare_IR_sensor Adafruit_motor_shield_v2
Adafruit_StepperMotor


#include <Wire.h>

#include <Adafruit_MotorShield.h> // library for
Adafruit_motor_shield_v2


// Create the motor shield object with the default I2C address

Adafruit_MotorShield AFMS1 = Adafruit_MotorShield(0x60);

// Create the motor shield on top of the AFMS1

Adafruit_MotorShield AFMS2 = Adafruit_MotorShield(0x61);


// Connect a stepper North with 200 steps per revolution (1.8 degree)
to motor port #2 (M1 and M2)

Adafruit_StepperMotor *MotorNorth = AFMS1.getStepper(200, 1);

// Connect a stepper South with 200 steps per revolution (1.8 degree)
to motor port #2 (M3 and M4)
```

```
Adafruit_StepperMotor *MotorSouth = AFMS1.getStepper(200, 2);

// Connect a stepper Feeder with 200 steps per revolution (1.8 degree)
to motor port #2 (M1 and M2)

Adafruit_StepperMotor *MotorFeeder = AFMS2.getStepper(200, 1);


#define ledPin 4 // for visualizing error

#define IRNorth 11 // changed to pin 11 as pin 13 is not working!

#define IRSouth 12 // IR sensor to pin 12


unsigned long Time1 = 0;

unsigned long Time2 = 0;

unsigned long TimeDifference = 0;

int count = 0; // execute the function only once

int starvingT = 30; // define starving time (in ms). If the animal did
not eat for this time, you will receive email notification

int waitingT = 30; // define waiting time (in ms). This is waiting
time between two trials in which the animal cannot enter the maze
(both doors closed)


// blink out an error code (number of blinks equal the error number)

void error(uint8_t errno) {

    while(1) {

    uint8_t i;

    for (i=0; i<errno; i++) {

      digitalWrite(ledPin, HIGH);

      delay(100);

      digitalWrite(ledPin, LOW);

      delay(100);

    }

    for (i=errno; i<10; i++) {

      delay(200);

    }

  }
```

```
}

void setup() {
  // Define IR sensor imput (digital)
  pinMode(IRNorth,INPUT);
  pinMode(IRSouth,INPUT);

  Serial.begin(9600);           // set up Serial library at 9600 bps
  AFMS1.begin();  // create with the default frequency 1.6KHz
  AFMS2.begin();  // create with the default frequency 1.6KHz
  MotorNorth->setSpeed(10);  // 10 rpm
  MotorSouth->setSpeed(10);  // 10 rpm
  MotorFeeder->setSpeed(10); // 10 rpm
 }

void loop() {

 //when booted, move motorSouth up
 if(count == 0) //
    {
      MotorSouth->step(12, FORWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      count = 1; //if executed, do not repeat
            }
     else{
        }

//Serial.println(count); //for troubleshooting

//when IRSouth activated, move motorNorth up and Feeder once
 if(digitalRead(IRSouth)==0 && count == 1 ) // digitalRead == 0
(circuit grounded) when the sensor is active
```

```
    {
      MotorNorth->step(12, FORWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      MotorFeeder->step(54, BACKWARD , SINGLE); // drop the food
      count = 2; //if executed, do not repeat
      }
     else{
        }


Serial.println(count); //for troubleshooting


//Serial.println(digitalRead(IRNorth)); //for troubleshooting


 //when IRNorth activated, move motorNorth down and motorSouth down.
After 20 min move motorSouth up.
 if(digitalRead(IRNorth)==0 && count == 2 ) // digitalRead == 0
(circuit open) when the sensor is active
    {
      Serial.println("IRNorth activated"); // send message to
Raspberry Pi when IRNorth is activated


      Time2 = millis(); // Time1, Time1 and TimeDifference used to
calculate ms elapsed between two activations of IRNorth
      TimeDifference = Time2 - Time1;
        if(TimeDifference > starvingT) { //3600000 is 1h
          Serial.println("Animal did not eat for 1h!"); // send
message to Raspberry Pi
        }
      Serial.print("Time difference between activating IRNorth is: ");
Serial.print (TimeDifference); Serial.println (" ms");
      MotorNorth->step(12, BACKWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      MotorSouth->step(12, BACKWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
```

```
      Time1 = millis();

      delay(waitingT); // wait, 1200000 ms is 20 min

      MotorSouth->step(12, FORWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps

      count = 1; //if executed, wait for IRSouth to get activated

      }

     else{

        }


}
```

Raspberry pi

```
// OS Windows 10, Arduino 1.8.10

// Author Lara Vrbanec

// Date 14/04/2020

// code sources Adafruit_motor_shield_v2
https://learn.adafruit.com/adafruit-motor-shield-v2-for-
arduino/stacking-shields

// sensors Waveshare_IR_sensor Adafruit_motor_shield_v2
Adafruit_StepperMotor


#include <Wire.h>

#include <Adafruit_MotorShield.h> // library for
Adafruit_motor_shield_v2


// Create the motor shield object with the default I2C address

Adafruit_MotorShield AFMS1 = Adafruit_MotorShield(0x60);

// Create the motor shield on top of the AFMS1

Adafruit_MotorShield AFMS2 = Adafruit_MotorShield(0x61);


// Connect a stepper North with 200 steps per revolution (1.8 degree)
to motor port #2 (M1 and M2)

Adafruit_StepperMotor *MotorNorth = AFMS1.getStepper(200, 1);

// Connect a stepper South with 200 steps per revolution (1.8 degree)
to motor port #2 (M3 and M4)
```

```
Adafruit_StepperMotor *MotorSouth = AFMS1.getStepper(200, 2);

// Connect a stepper Feeder with 200 steps per revolution (1.8 degree)
to motor port #2 (M1 and M2)

Adafruit_StepperMotor *MotorFeeder = AFMS2.getStepper(200, 1);


#define ledPin 4 // for visualizing error

#define IRNorth 13 // IR sensor to pin 13

#define IRSouth 12 // IR sensor to pin 12


unsigned long Time1 = 0;

unsigned long Time2 = 0;

unsigned long TimeDifference = 0;

int count = 0; // execute the function only once


// blink out an error code (number of blinks equal the error number)

void error(uint8_t errno) {

    while(1) {

    uint8_t i;

    for (i=0; i<errno; i++) {

      digitalWrite(ledPin, HIGH);

      delay(100);

      digitalWrite(ledPin, LOW);

      delay(100);

    }

    for (i=errno; i<10; i++) {

      delay(200);

    }

  }

}


void setup() {
  // Define IR sensor imput (digital)
  pinMode(IRNorth,INPUT);
  pinMode(IRSouth,INPUT);

  Serial.begin(9600);           // set up Serial library at 9600 bps
  AFMS1.begin();  // create with the default frequency 1.6KHz
  MotorNorth->setSpeed(10);  // 10 rpm
  MotorSouth->setSpeed(10);  // 10 rpm
  MotorFeeder->setSpeed(10); // 10 rpm
 }

void loop() {

 //when booted, move motorSouth up
 if(count == 0) //
    {
      MotorSouth->step(50, BACKWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      count = 1; //if executed, do not repeat
      }
     else{
        }

 //when IRSouth activated, move motorNorth up and Feeder once !add
feeder
 if(digitalRead(IRSouth)==0 && count == 1 ) // digitalRead == 0
(circuit grounded) when the sensor is active
    {
      MotorNorth->step(50, BACKWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      MotorFeeder->step(10, FORWARD, SINGLE); // drop the food
      count = 2; //if executed, do not repeat
      }
     else{
        }
 //when IRNorth activated, move motorNorth down and motorSouth down.
After 20 min move motorSouth up.
 if(digitalRead(IRNorth)==0 && count == 2 ) // digitalRead == 0
(circuit open) when the sensor is active
    {
      Serial.println("IRNorth activated"); // send message to
Raspberry Pi when IRNorth is activated


      Time2 = millis(); // Time1, Time1 and TimeDifference used to
calculate ms elapsed between two activations of IRNorth
      TimeDifference = Time2 - Time1;
        if(TimeDifference > 3600000) { //3600000 is 1h
          Serial.println("Animal did not eat for 1h!"); // send
message to Raspberry Pi
        }
      Serial.print("Time difference between activating IRNorth is: ");
Serial.print (TimeDifference); Serial.println (" ms");
      MotorNorth->step(50, FORWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      MotorSouth->step(50, FORWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      Time1 = millis();
      delay(1200000); // wait, 1200000 ms is 20 min
      MotorSouth->step(50, BACKWARD, SINGLE); // 50 is 90 degrees as
the motor has 200 steps
      count = 1; //if executed, wait for IRSouth to get activated
      }
     else{
        }


}
```

Procedure for animal testing

In order to test exploration related to food search, animal is prior to testing food deprived for two hours (Keicher, 2017). The testing is conducted in dim light (< 10 lux) as it is believed that shrews are the most active in dawn and twilight. The trial starts when the home cage is connected with the maze with a transparent tube.

Animals are tested in multiple T maze for 12 trials in a row, with maximum of 6 trials during a single day. If the animal does not participate in the trial within 1.5 h from the start of the experiment, it is returned to the outside enclosure. If this occurs for three days, animal is excluded from the experiment and marked as not participating. If the animal does not discover the goal arm within 1h, it gently chased to the home cage and provided a piece of mealworm. The sand is replaced between different subjects and maze walls are cleaned with disinfectant.

The trial starts when the animal voluntarily enters the maze from a home cage, which activates automatic food dispenser that provides a mealworm in the goal arm of the maze. The trial is over when the animal reaches the goal arm, after which the automatic doors are closed and the animal is prevented from re-entering the maze for 20 min. After 20 min only the doors at the start of the maze open, preventing the animal to explore the maze from the goal arm. The next trial starts when the animal enters the start arm of the maze, which again activates the food dispenser and opens the doors at the goal end of the maze. Animal movement is tracked with the IR surveillance camera connected to the cloud storage over institute Ethernet.

Analysis

Video recordings are analyzed with automatic tracking software to obtain animals' movement coordinates. From the movement data we extract the number of wrong turns for each of the trials, the total distance covered during a single trial, and the time spent active in the maze during a single trial, as well as the sequence of turns (e.g. L L L R R …) for the first trial.

The number of wrong turns over multiple trials, total distance covered over multiple trials, and the time spent actively exploring the maze over multiple trials are used to a design learning curves.
The sequence of turns during the first trial is used as a proxy for non-cognitive strategy and is compared to the results of non- cognitive strategy obtained in a Y maze.