Using Robotbit to Simulate the Behaviour and Locomotion of a Moth

Gabrielle Branche

New York University Shanghai

geb297@nyu.edu

Abstract

This experiment covered the creation of a robotic moth that could respond to its environment by means of light sensors, ultrasound, servos and IR sensors. By using concepts of biomimetics and Braitenberg's creatures, the moth was assembled using a kittenbot and microbitsimulationg both the behavioral and locomotive characteristics of a moth. This technology is a rudimentary form of concepts that can be applied to the creation of larger bio-inspired robotics that can be used in better understanding nature.

Introduction:

Biomimetics, the process of applying and deriving nature in technological designs has the ability to inform large technological advances (Bhushan, 2009). The use of biomimetics can be used to develop the field of robotics particularly by designing bio-inspired robots. In order to accurately apply biomimetics in robotics it is important to understand both the behaviour and the mechanics of the organisms being simulated.

The moth has very distinct characteristics both behaviorally and mechanically that make it an appropriate organism to simulate. The characteristics of the moth are important to their survival and as such qualities pertaining to mating, movement and flight play a vital role in their design. Additionally, their response to light is a trademark characteristic that can be simulated. By designing a robot that emulates the behaviour and mechanics of a moth the organism can be better understood and used as a template to develop other larger organisms.

This paper discusses the process of designing a robotic moth using microbit to code it's behaviour in conjunction with its locomotion.

Materials and Methods:

The moth was created by using the skeleton of a a (microbit) kittenbot. The ultrasound eyes were used to detect obstacles and mating sites. The IR sensor at the bottom of the bot was used to detect the border that kept the moth confined. This was used to simulate the hive in which the female moths spent most of the time. A light sensor was attached to the bot to trap the moth in intense light. Extra servos were added on either side of the bot and used in conjunction with fabricated wings to simulate the flapping movement of a moth. Finally although the moth itself could not fly due to aerodynamic constraints, wheels were used to allow the moth

the move around its environment and respond to stimulus.

Results:

The moth created was able to follow the following logic:

- Move randomly across the space ‘ovipositing eggs’ (represented by green neopixels)
- If the moth reaches the border stop turn and move in the opposite direction
- If the moth runs out of eggs, move towards a mating sight and collect eggs (represented by pink neopixels) then turn away
- If the moth is exposed to light circle around it until the light is removed or the moth dies

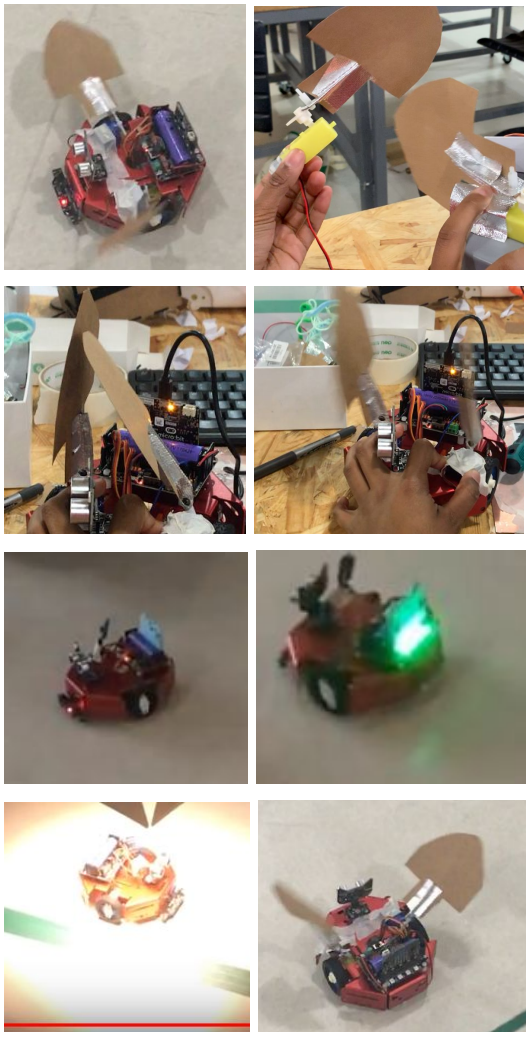

Figure 1: The Kittenbot at different stages of the simulation from different angles

Discussion:

Braitenberg’s creatures refer to a collection of behaviors that emulate simple responses experienced by a majority of living organisms. One of these creatures is called Paranoid. Paranoid’s mechanism was that of getting trapped in a loop of wanting to explore but then getting scared. This continuum created around the light is similar to that of a moth that once stuck in the light does not stray away from it and then eventually dies. The robot was made to not be attracted to the light but simply getting trapped when exposed to it. This is due to the fact the moths the reason for moth’s attraction is unknown. SOme believe that the light is mistaken for the moon (Langevelde et al, 2011). However it is a phenomenon that is highly debated. For the purpose of this experiment, only the observable was used and since we could not clarify the behaviour of the moth, attraction to the light can also not be justified.

By using a variety of sensors, a holistic approach to creating the microbot could happen. The Greater Wax Moth, one of the moths used as a reference point for building this model, mates at least once a day (Nielson, 1977). As such most of the sensors, minus the light sensor, is focused on navigating the space such that eggs are widely spread and the robot could find its way to other mating sites.

Due to the limited range of the ultrasound, it was important to ensure that obstacles were wisely placed such that the kittenbot had enough time to sense them and respond accordingly. Ideally the values would be read via webcam and the position of the obstacles and mating sites could be read remotely and sent to the bot via serial and radio communication. (HOwever due to time constraints the experiment was limited to only local sensors which made it difficult to manipulate the moth’s ability to asses the objects around it.

While the robot could not explicitly fly, it was important to assess the locomotive characteristics of a moth which is largely defined by symmetrical wing movement. Wings need to have some level of flexibility to allow for a fluid movement of flapping (Smith, 1996). By using a servo and an angular setup with the wings the movement was made more fluid and more accurately depicted the flapping of a moth.

Additionally, in a study of *Hawkmoth* flight, it was indicated that the angle of rotation of the wings were not constant and changes with speed and flight distance (Willmont and Ellington 1997). As such the change in the angle of the servo was randomized. However, in line with biomimetics as the moth carried out various functions such as mating and being trapped in light, the speed of flapping was manipulated to match that of observation.

The female greater wax moth is known to remain in the hive after fertilization while the male and virgin female moths go out in the early morning and night (Nielson, 1977). This was represented by the box that constrained the bot. A further development to this project would be to have both male and female bots in the hive with the former being able to roam freely and the latter only exiting the box if they are not fertilized.

The biggest challenge of this experiment was ensuring that the various sections of code did not prevent the other sections to be read in time. To solve this problem functions were made which allowed for code to only be read when needed.

This experiment was an apt way to explore biomimetics, animal behaviour and locomotion. However, it can still be largely be improved both in design and efficiency. With smaller equipment, the weight can be balanced to a point that perhaps the moth could actually fly rather than relying on wheels.

Conclusion:

There is a great deal to learn from observing and imitating natural phenomena. Bio-inspired robotics has the potential to change our understanding of the natural world and integrate technology and biology in way up until now only imaginable. By starting with small organisms such as the moth and using theories such as Braitenberg's creatures to understand the behaviour and mechanics of organisms, bigger more complex designs can then be created. Through observation and research, it was possible to create a moth that simulated the behaviour and locomotion of a female Greater Wax Moth in a controlled environment.

References:

Gopalakrishnan, Pradeep, and Danesh K. Tafti. "Effect Of Wing Flexibility On Lift And Thrust Production In Flapping Flight". *AIAA Journal*, vol 48, no. 5, 2010, pp. 865-877. *American Institute Of Aeronautics And Astronautics (AIAA)*, doi:10.2514/1.39957.

Lilienthal, A. and Duckett, T. (2004). Experimental analysis of gas-sensitive Braitenberg vehicles. *Advanced Robotics*, 18(8), pp.817-834.

Nielsen, R. and Brister, D. (1977). The Greater Wax Moth:1 Adult Behavior2. *Annals of the Entomological Society of America*, 70(1), pp.101-103.

Smith, Michael J. C. "Simulating Moth Wing Aerodynamics - Towards The Development Of Flapping-Wing Technology". *AIAA Journal*, vol 34, no. 7, 1996, pp. 1348-1355. *American Institute Of Aeronautics And Astronautics (AIAA)*, doi:10.2514/3.13239. Accessed 5 May 2019.

Willmott, Alexander P., and Charles P. Ellington. "The Mechanics of Flight in the Hawkmoth Manduca Sexta" *The Journal of Experimental Biology, vol 200, 1997, pp. 2705-2722.,* Accessed 5 May 2019.

Appendix 1: Master Code

```
# Write your code here :-)
from microbit import *
import robotbit
import time
import random
import neopixel

# Setup the Neopixel strip on pin0 with
a length of 8 pixels
np = neopixel.NeoPixel(pin16, 4)

np[0] = (0, 255, 0)
np[1] = (0, 255, 0)
np[2] = (0, 255, 0)
np[3] = (0, 255, 0)
np.show()
sleep(random.randint(0, 3)*1000)

def fly(speed):
robotbit.servo(0, 30)
robotbit.servo(1, 150)
time.sleep(speed)
robotbit.servo(0, 150)
robotbit.servo(1, 30)
time.sleep(speed)


def light():
#light sensor
robotbit.motor(1, -105, 0)
robotbit.motor(3, 10, 0)
fly(200)
delay(6000)
robotbit.motor(1, 0, 0)
robotbit.motor(3, 0, 0)


def hive():
robotbit.motor(1, 100, 0)
robotbit.motor(3, 100, 0)
sleep(2000)
robotbit.motor(1, -105, 0)
robotbit.motor(3, 10, 0)
sleep(random.randint(0, 5)*1000)


def mating():
robotbit.motor(1, 0, 0)
robotbit.motor(3, 0, 0)

oviposit()

robotbit.motor(1, 100, 0)
robotbit.motor(3, 100, 0)
sleep(500)
robotbit.motor(1, -105, 0)
robotbit.motor(3, 10, 0)
sleep(random.randint(0, 3)*1000)

def oviposit():
np[0] = (0, 0, 0)
np[1] = (0, 0, 0)
np[2] = (0, 0, 0)
np[3] = (0, 0, 0)
np.show()
sleep(500)

np[0] = (255, 0, 128)
np.show()
sleep(1000)

np[1] = (255, 0, 128)
np.show()
sleep(1000)

np[2] = (255, 0, 128)
np.show()
sleep(1000)

np[3] = (255, 0, 128)
np.show()
sleep(1000)

np[0] = (0, 255, 0)
np[1] = (0, 255, 0)
np[2] = (0, 255, 0)
np[3] = (0, 255, 0)
np.show()

def roam():
robotbit.motor(1, -95, 0)
robotbit.motor(3, -90, 0)

fly(400)

sleep(random.randint(0, 10)*100)

np[3] = (0, 0, 0)
np.show()
sleep(random.randint(0, 20)*100)

np[2] = (0, 0, 0)
np.show()
sleep(random.randint(0, 20)*100)

np[1] = (0, 0, 0)
np.show()
sleep(random.randint(0, 20)*100)

np[0] = (0, 0, 0)
np.show()
# sleep(random.randint(0, 10)*100)


while True:
border = pin1.read_analog() #determine
light frequency
dist = robotbit.sonar(pin0) #determine
object distance
```

```
light  =  pin2.read_analog()  #determine
light intensity

#light sensor
if light < 50: #bright
light()

#IR Sensor
elif hive < 200: #black
border()

#ultrasound sensor
elif dist < 50 and dist != 0:
mating()

else: #ovipositing
roam()
```