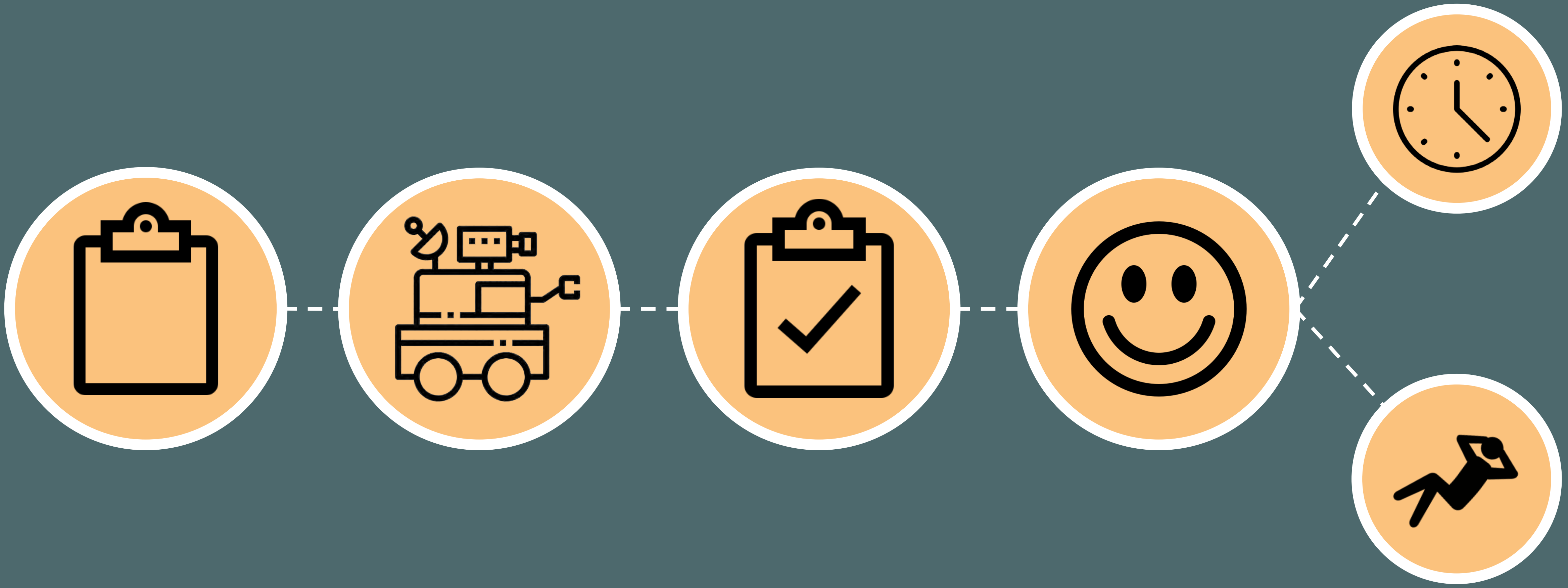
RECONFIGURING LABOUR

Towards autonomous robots for a moving desk office environment

Aaron Djong with BVN

HANDLING TASKS

GIVE THE TASK TO THE ROBOT



GIVE THE TASK TO THE ROBOT

BETTER OPERATIONS AND USE OF TIME

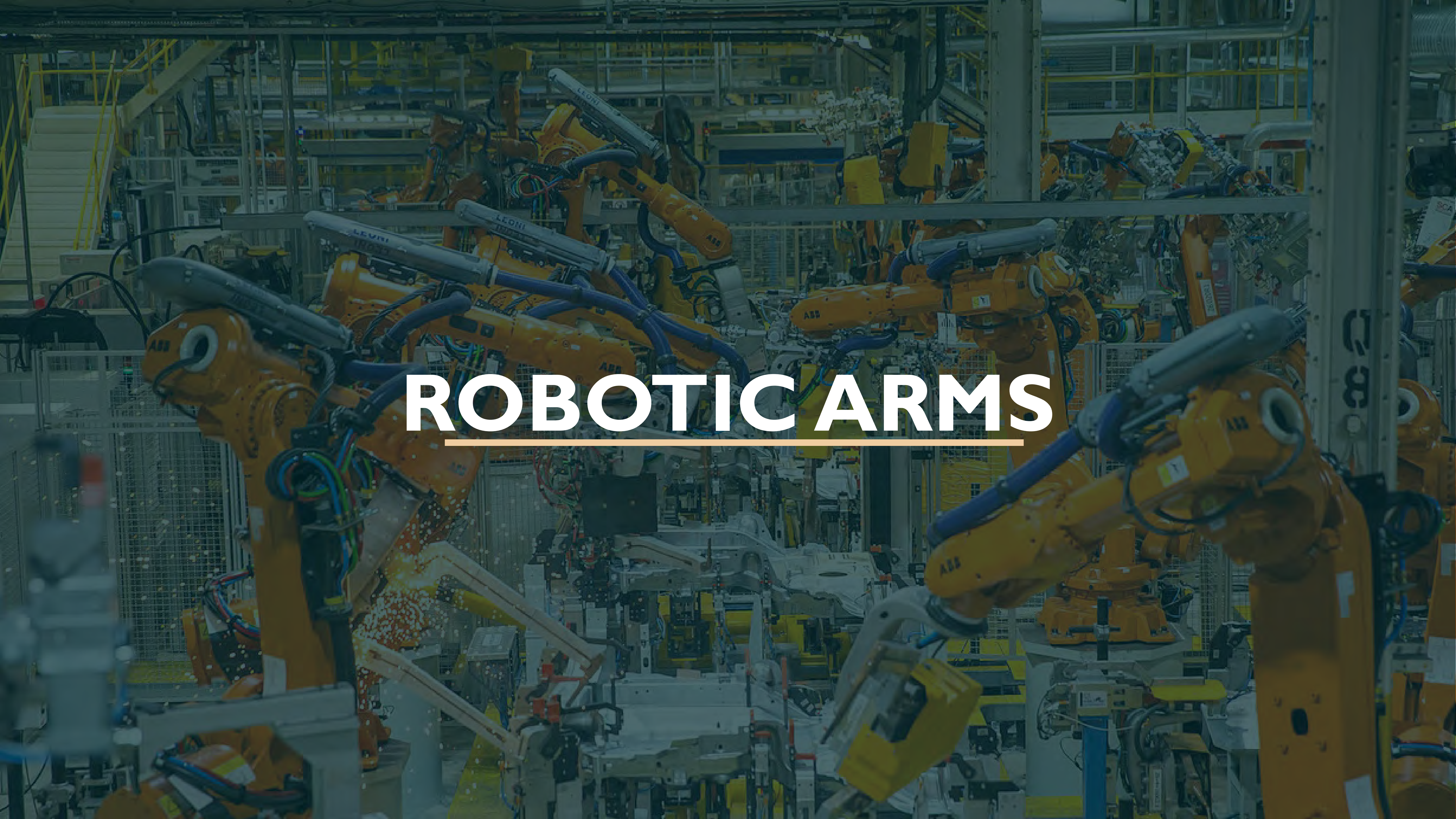
ROBOTS IN THE WORKPLACE

ACCURAT
E

FAST

CONSISTEN
T

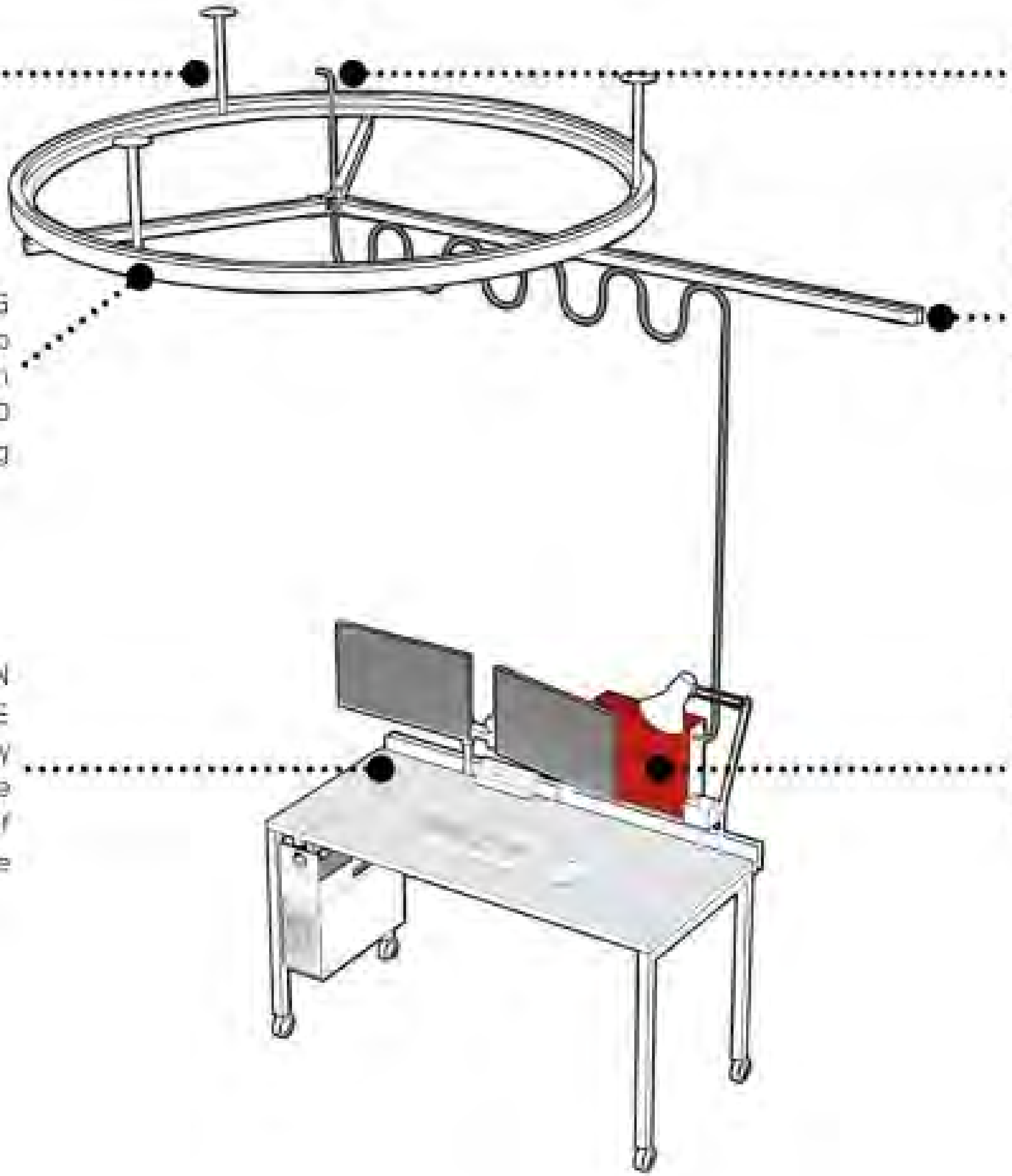
ROBOTIC ARMS



AMAZON WAREHOUSE ROBOTS

EXIT
205
346-104
ONE WAY
207
WRONG WAY
209
346-104
ONE WAY
210
211
WRONG WAY
212
213
WRONG WAY
214
WRONG WAY
CAUTION
0083
0028
0422
MONSTER
T-8
Deka

BUT



PATENT PENDING ©

HARNESS THE BENEFITS OF FIBRE OPTICS

THE DATA BOOM

Avoid the constraints of an umbilical. The 350 degree rotating boom allows you to connect your octopus anywhere within a 2m radius.

INTEGRATED LIGHTING

Avoid glare and add interest to the floorplate with integrated and addressable LED lighting

THE MOBILE WORKSTATION SYSTEM OF YOUR CHOICE

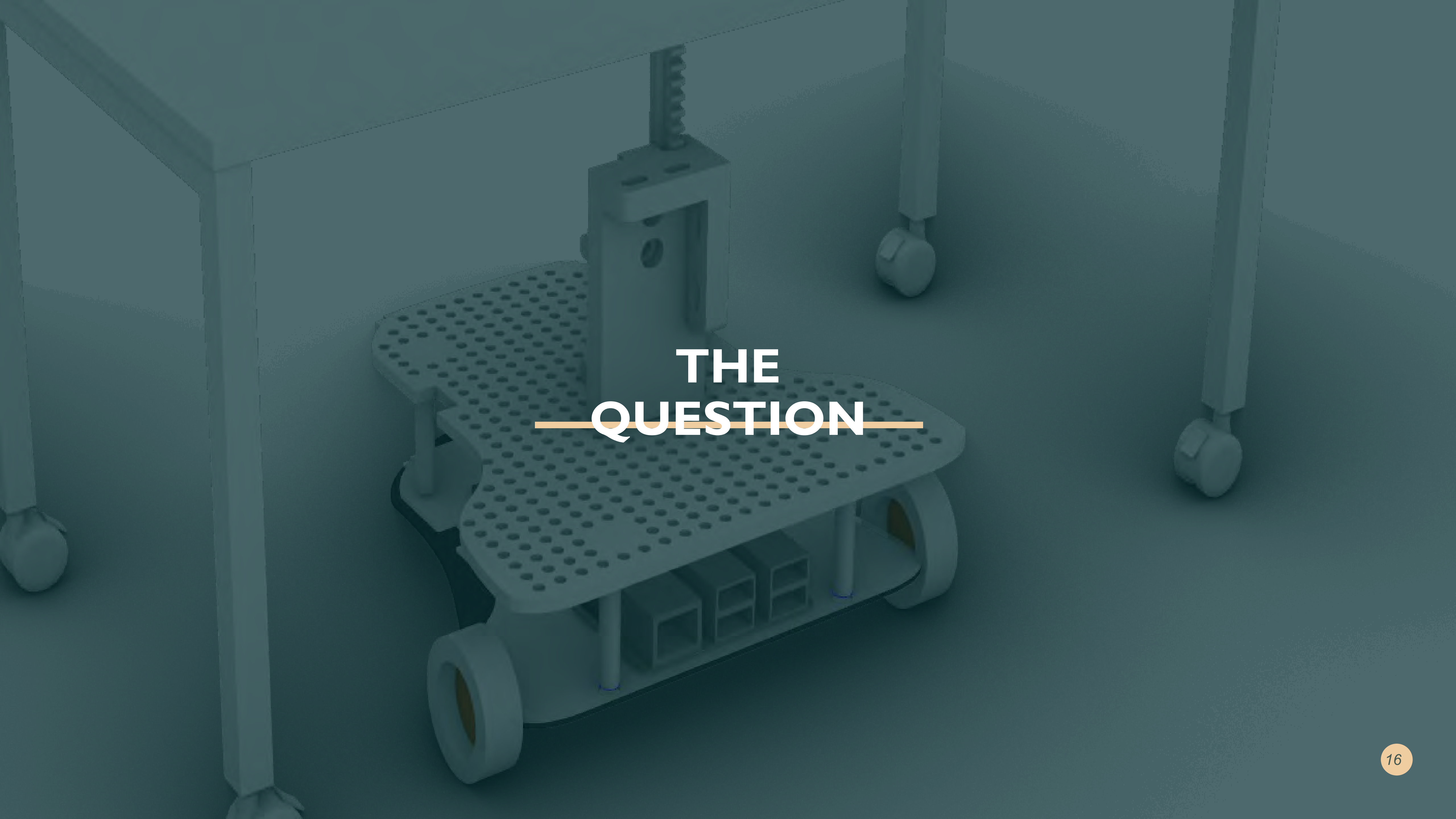
The Octopus can be easily configured to interface with the mobile workstation system of your preference

THE OCTOPUS

A plug and play system to allow users to connect up to 8 workstations to a cluster.

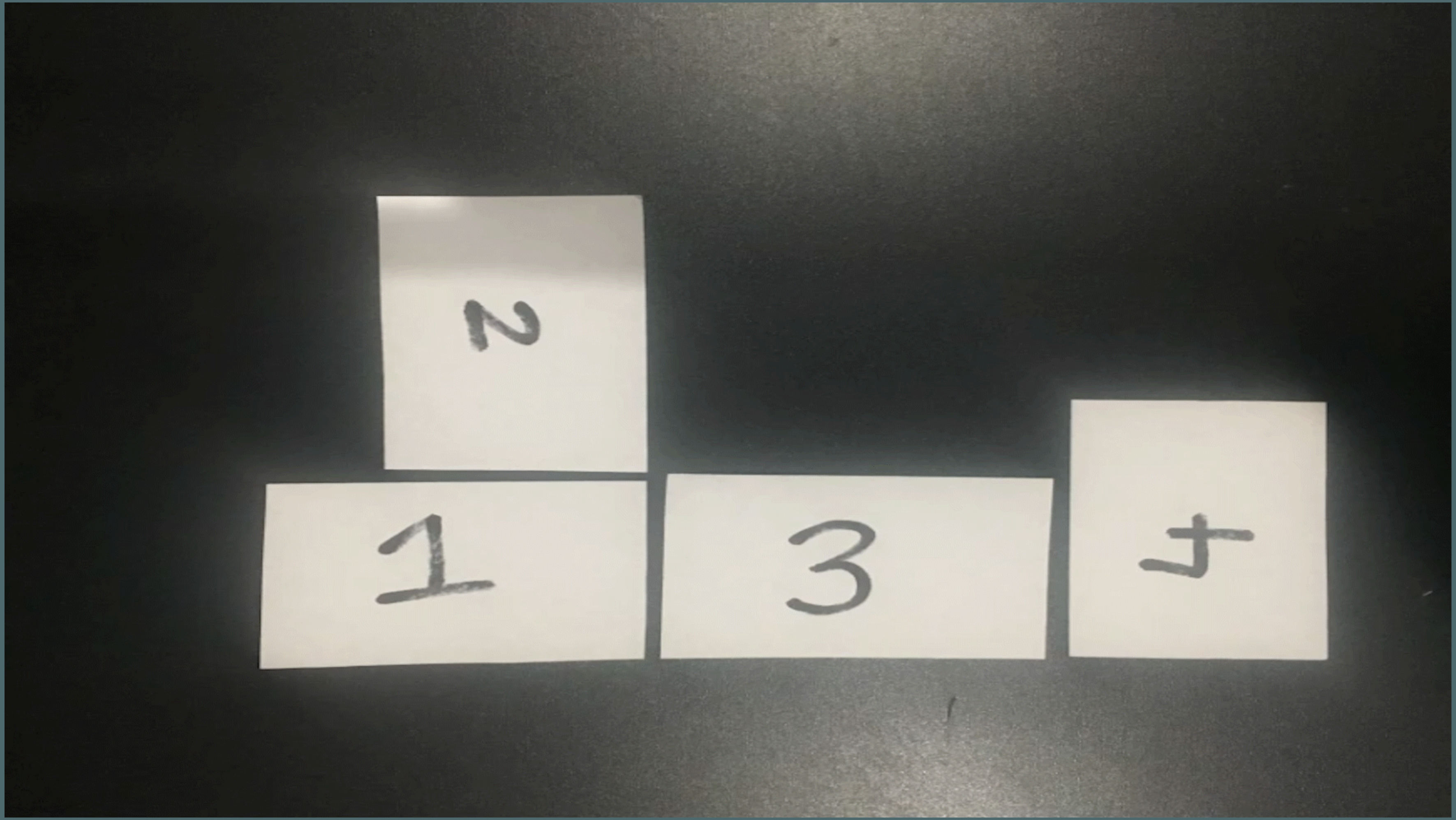
No electricians required!

THE PROBLEM

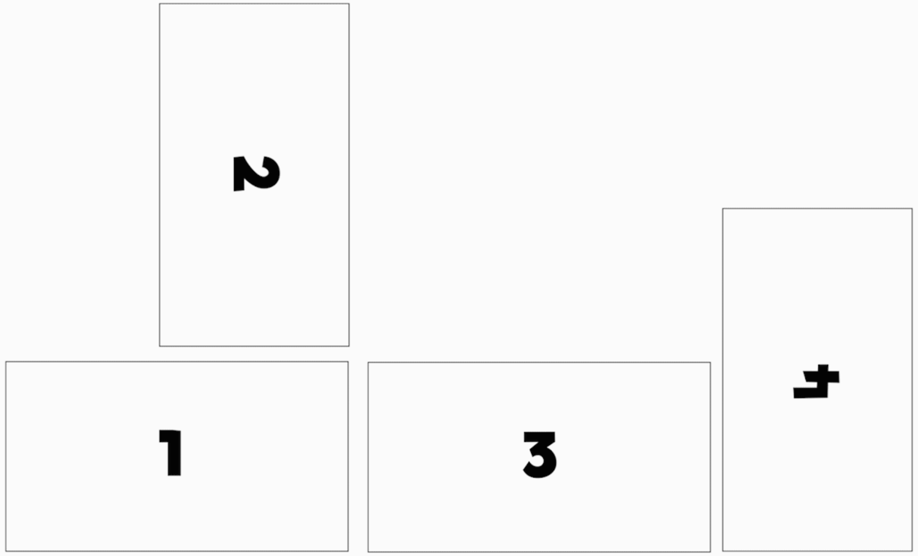


THE QUESTION

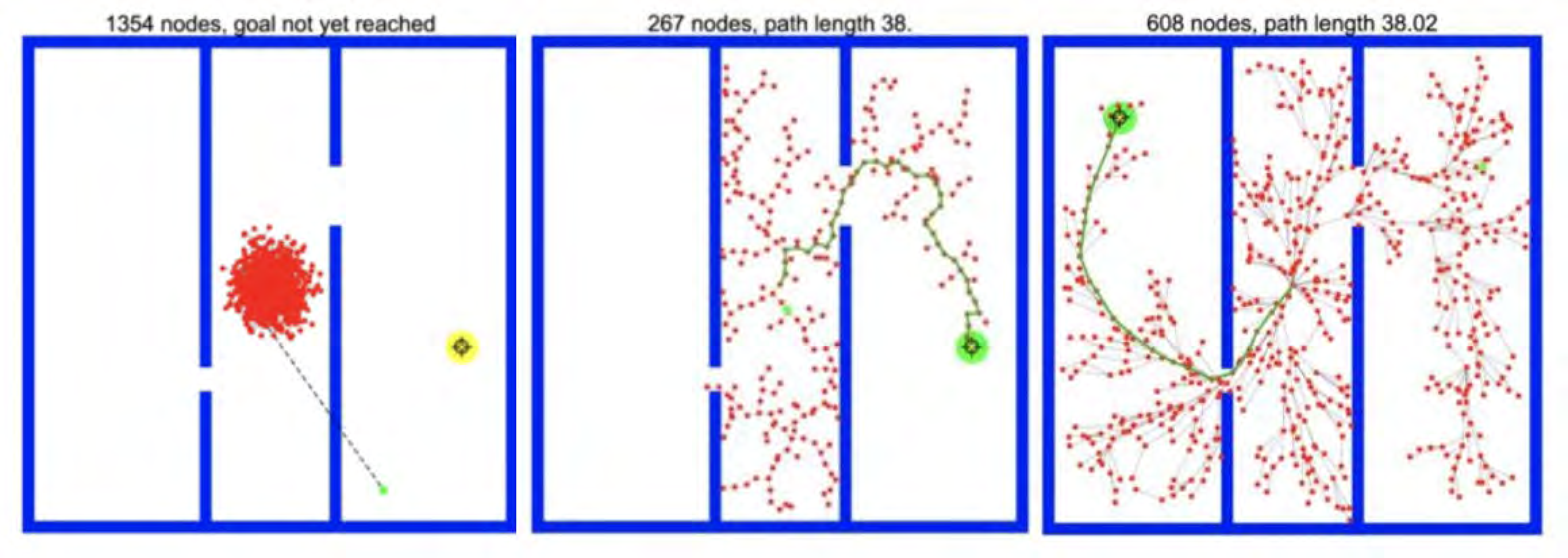
STOP MOTION MOVEMENT



DIGITAL RECREATION MOVEMENT



MOTION PLANNING ALGORITHMS


1354 nodes, goal not yet reached
267 nodes, path length 38.
608 nodes, path length 38.02

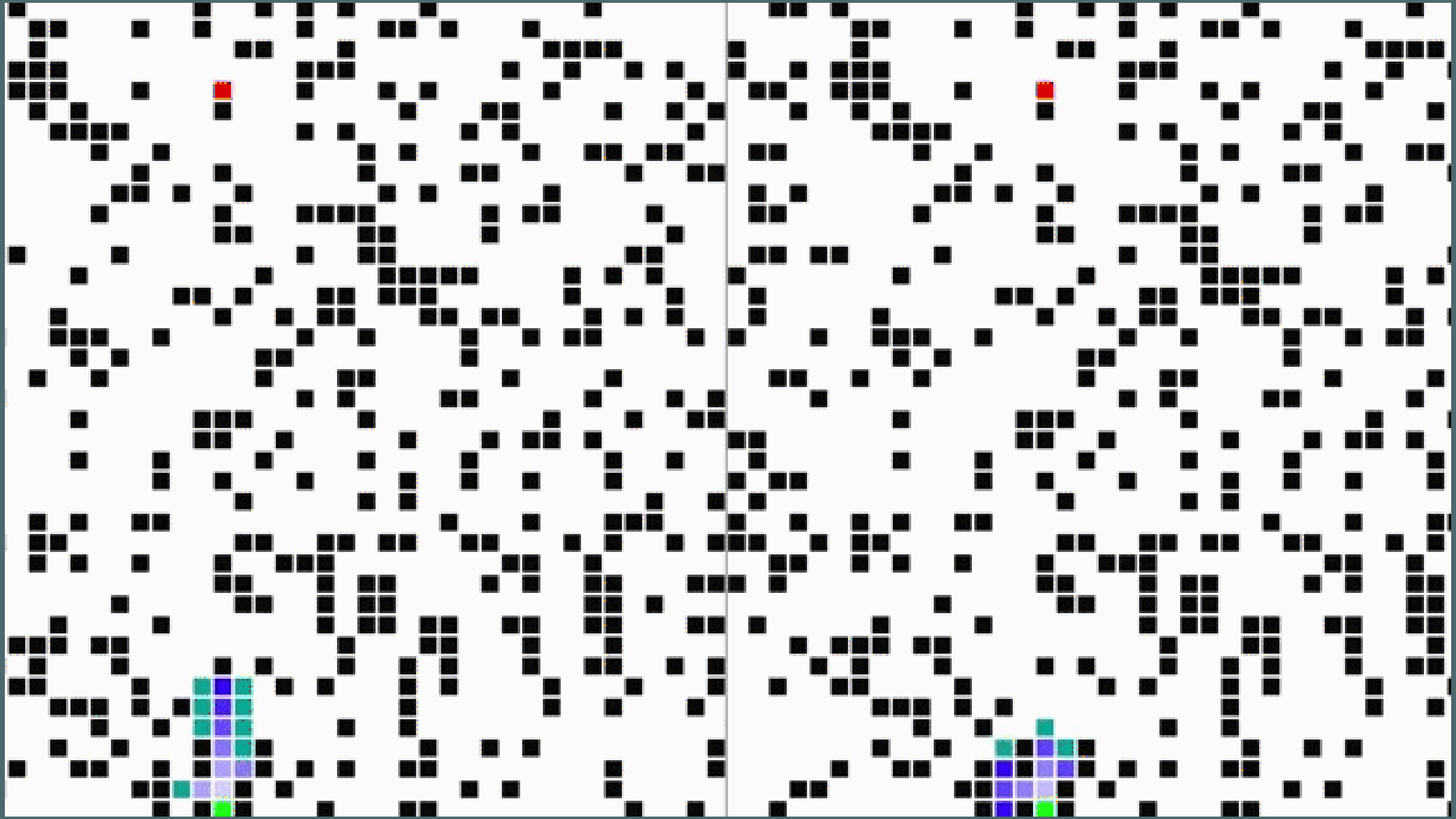
Random Tree (RT)

Rapidly Exploring Random Tree (RRT)

Rapidly Exploring Random Tree* (RRT*)

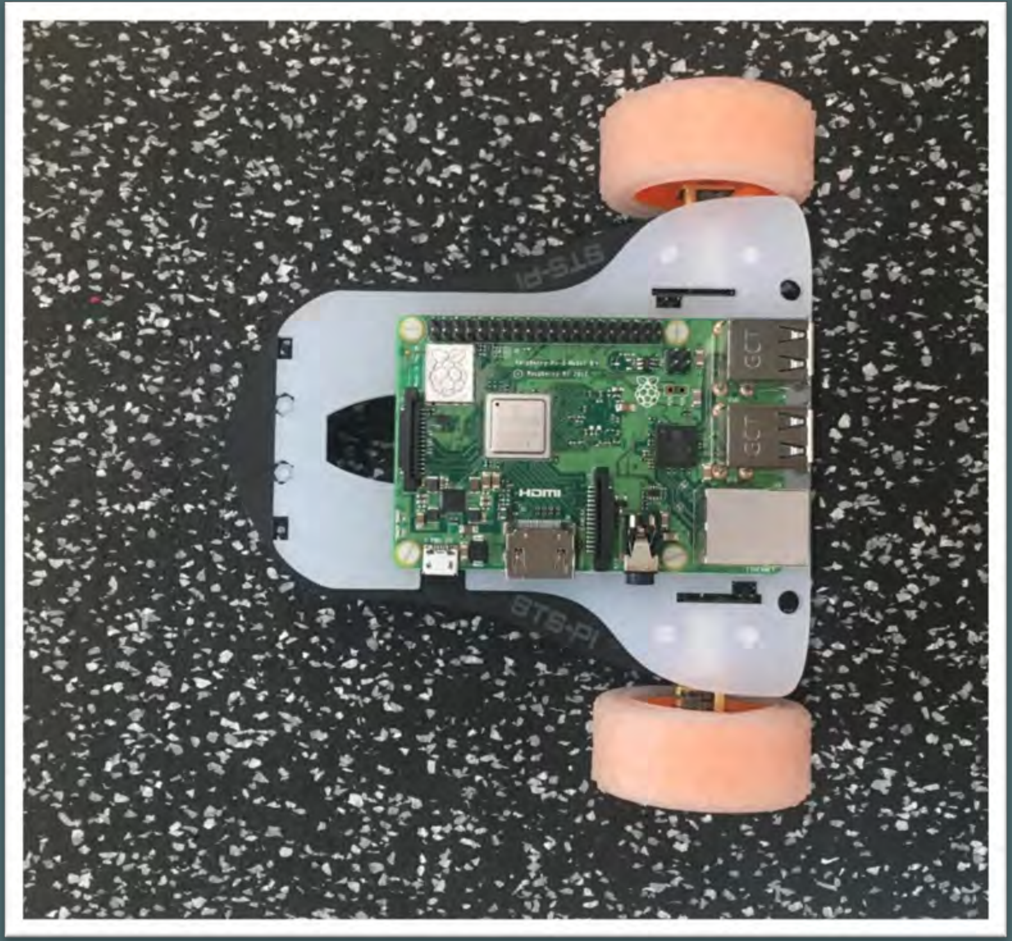
MOTION PLANNING ALGORITHMS

A*

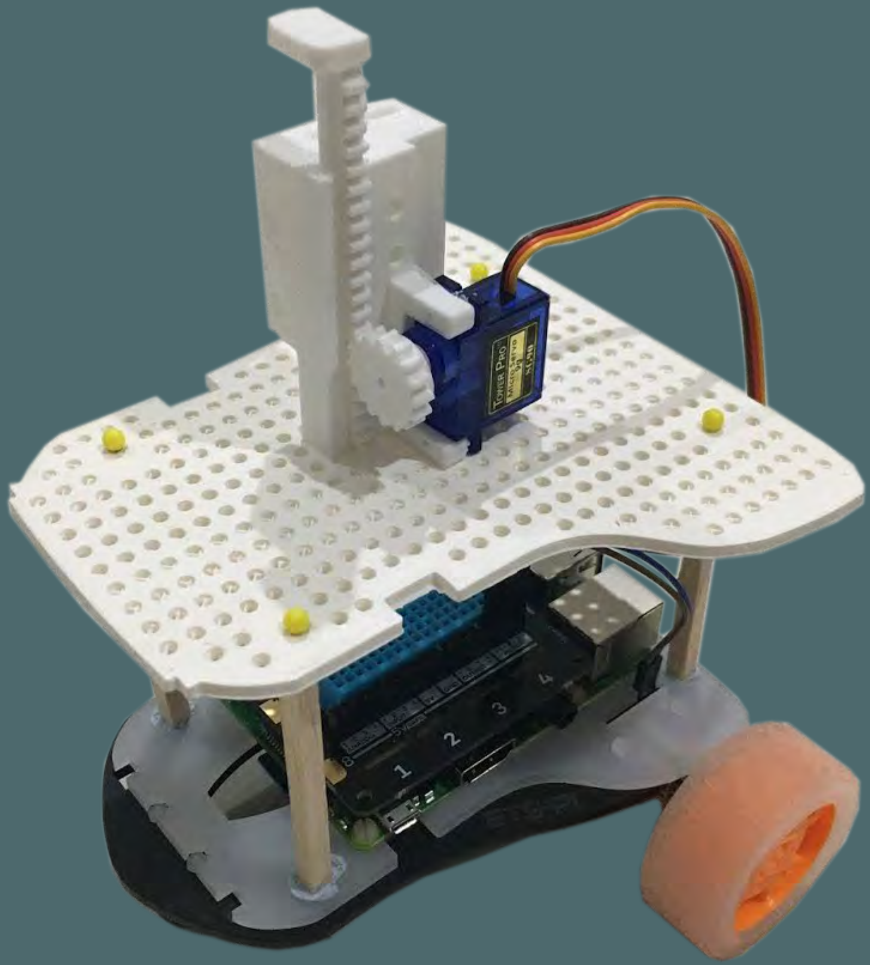


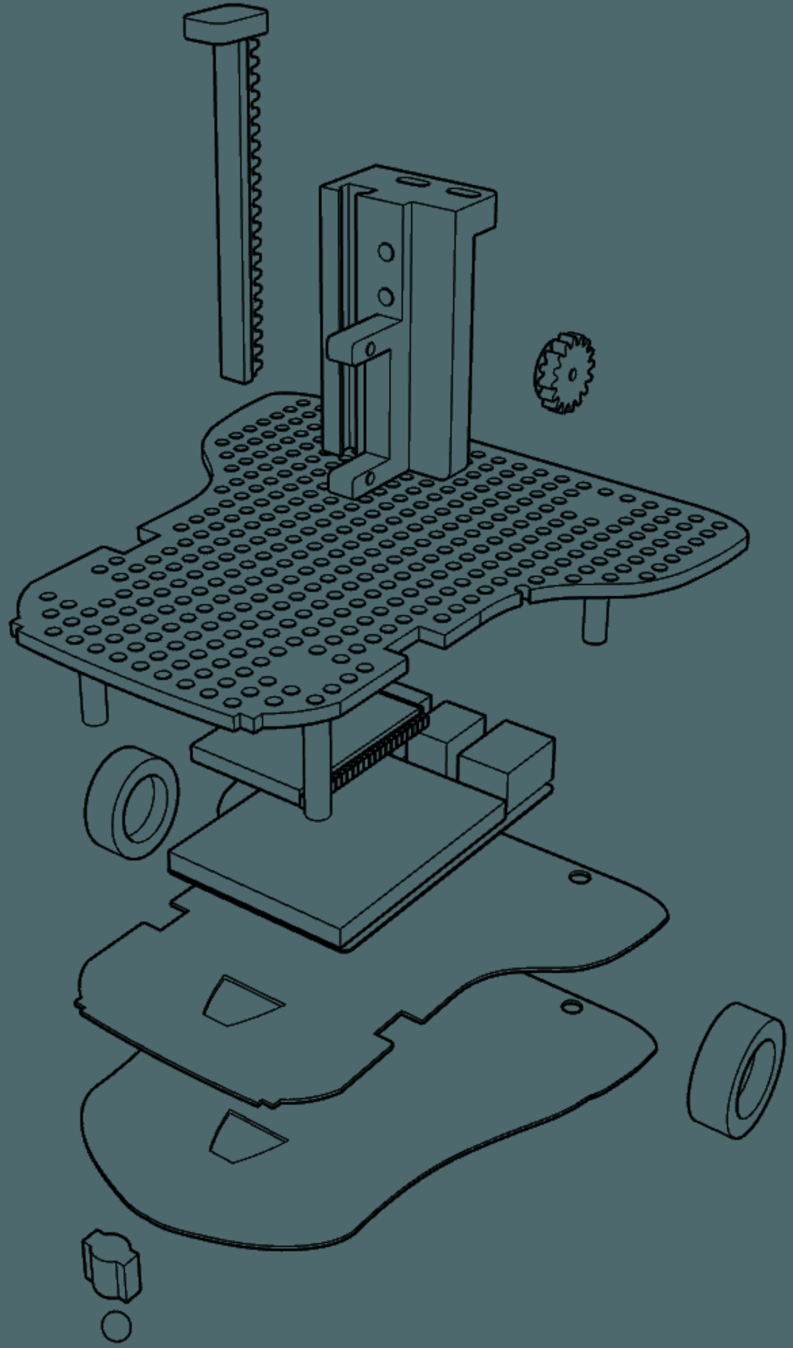
Dijkstra's

THE BASE ROBOT KIT: STS-PI

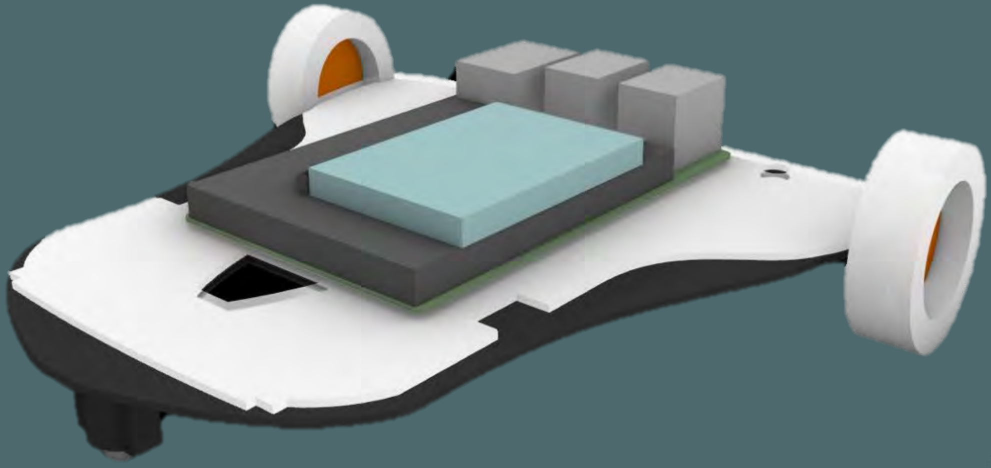


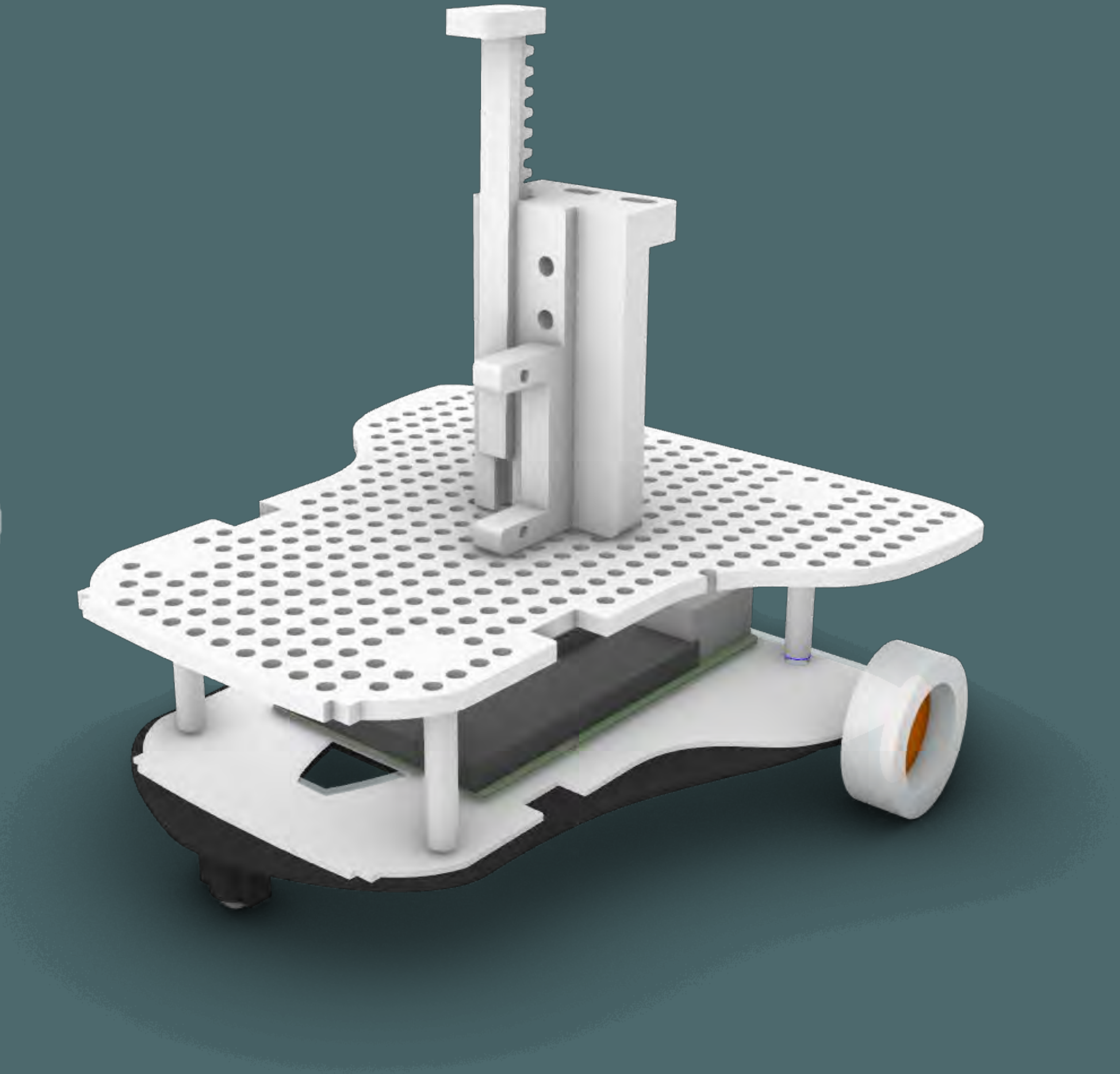
CUSTOMISED



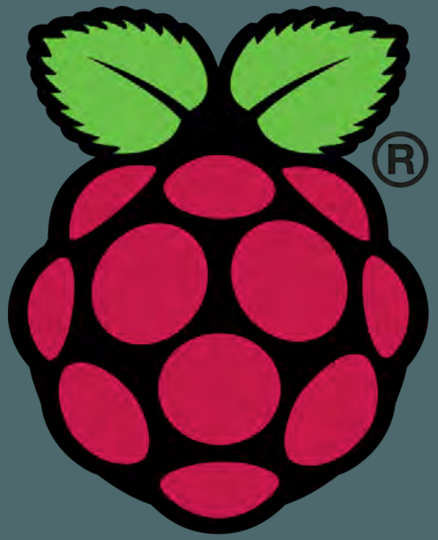


DIGITAL MODEL

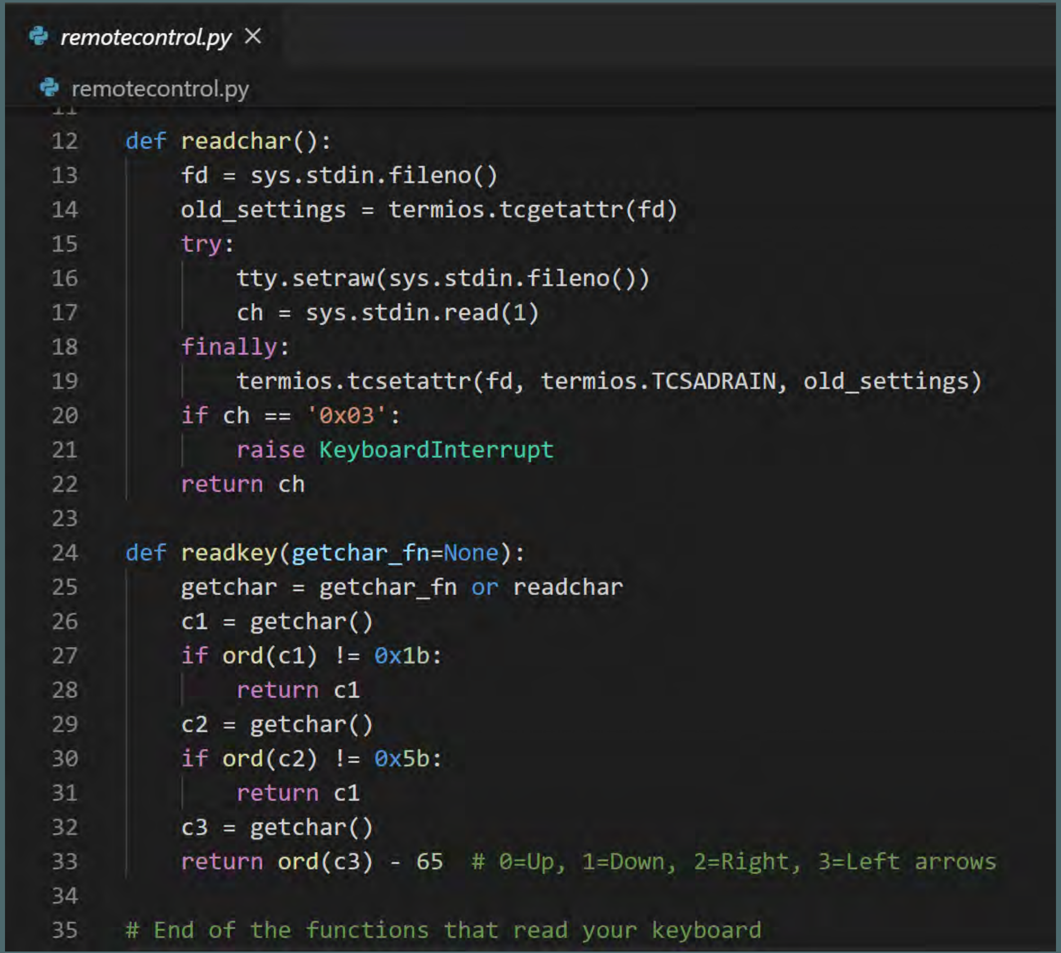




PROGRAMMING

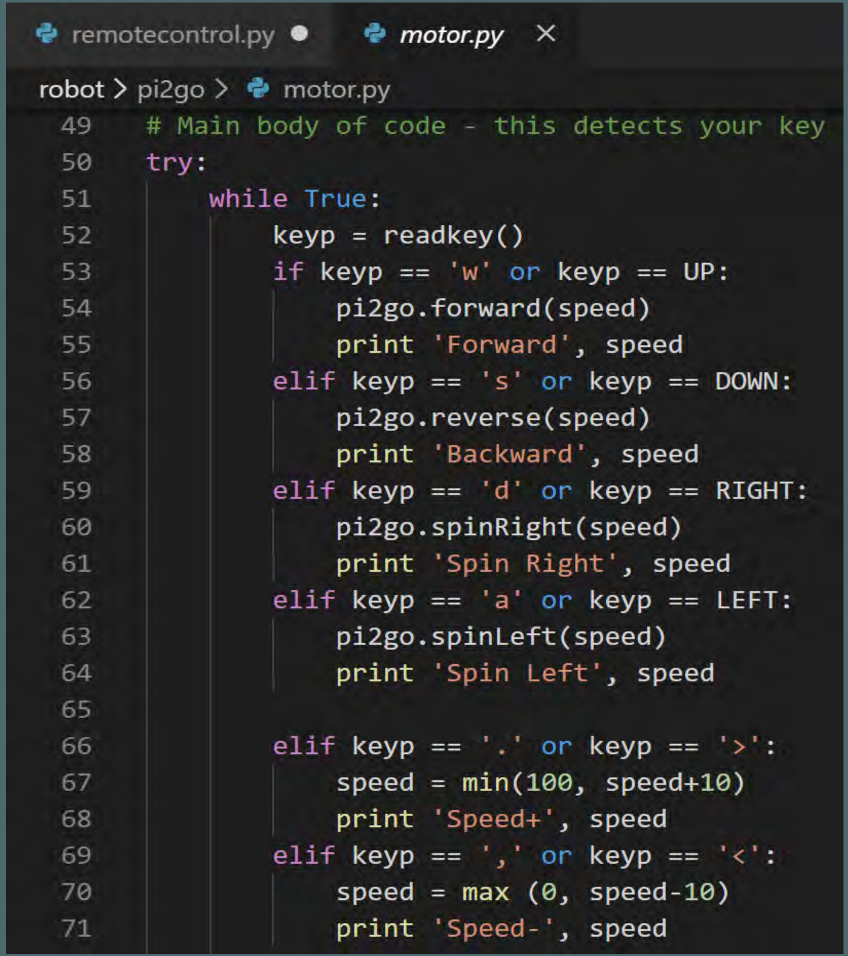


PROGRAMMING



```python
def readchar():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    if ch == '0x03':
        raise KeyboardInterrupt
    return ch

def readkey(getchar_fn=None):
    getchar = getchar_fn or readchar
    c1 = getchar()
    if ord(c1) != 0x1b:
        return c1
    c2 = getchar()
    if ord(c2) != 0x5b:
        return c1
    c3 = getchar()
    return ord(c3) - 65  # 0=Up, 1=Down, 2=Right, 3=Left arrows

# End of the functions that read your keyboard
```

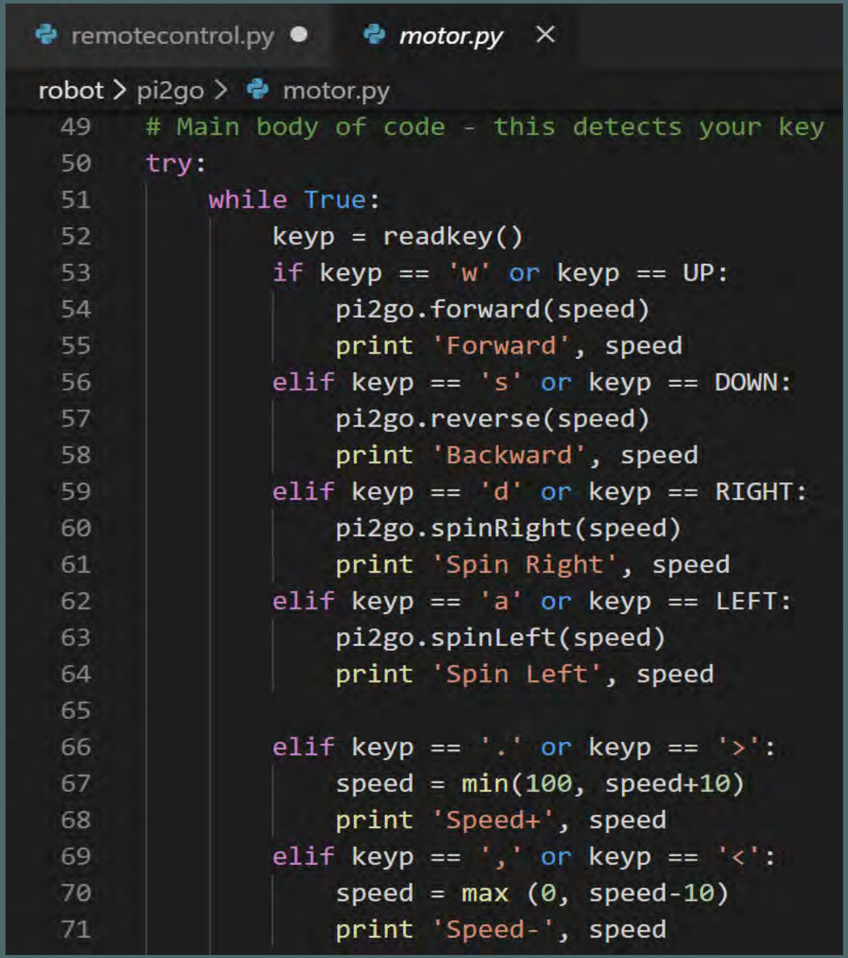
Keyboard Key Reader



```python
# Main body of code - this detects your key
try:
    while True:
        keyp = readkey()
        if keyp == 'w' or keyp == UP:
            pi2go.forward(speed)
            print 'Forward', speed
        elif keyp == 's' or keyp == DOWN:
            pi2go.reverse(speed)
            print 'Backward', speed
        elif keyp == 'd' or keyp == RIGHT:
            pi2go.spinRight(speed)
            print 'Spin Right', speed
        elif keyp == 'a' or keyp == LEFT:
            pi2go.spinLeft(speed)
            print 'Spin Left', speed

        elif keyp == '.' or keyp == '>':
            speed = min(100, speed+10)
            print 'Speed+', speed
        elif keyp == ',' or keyp == '<':
            speed = max (0, speed-10)
            print 'Speed-', speed
```

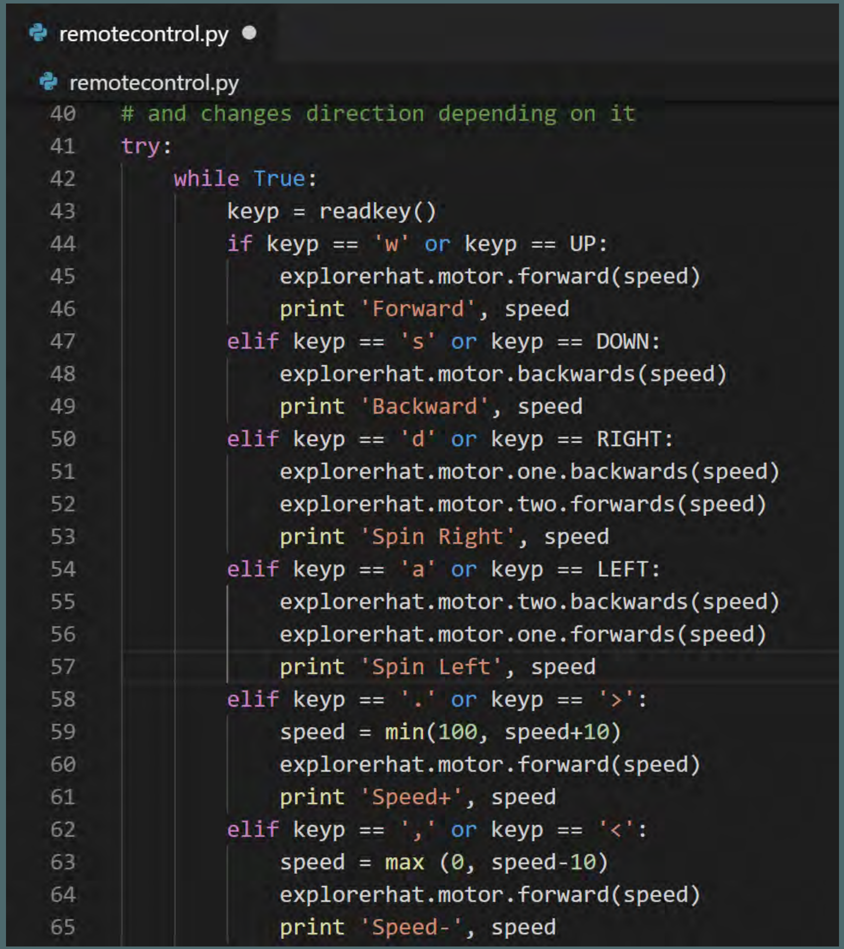
Motor Controls

PROGRAMMING



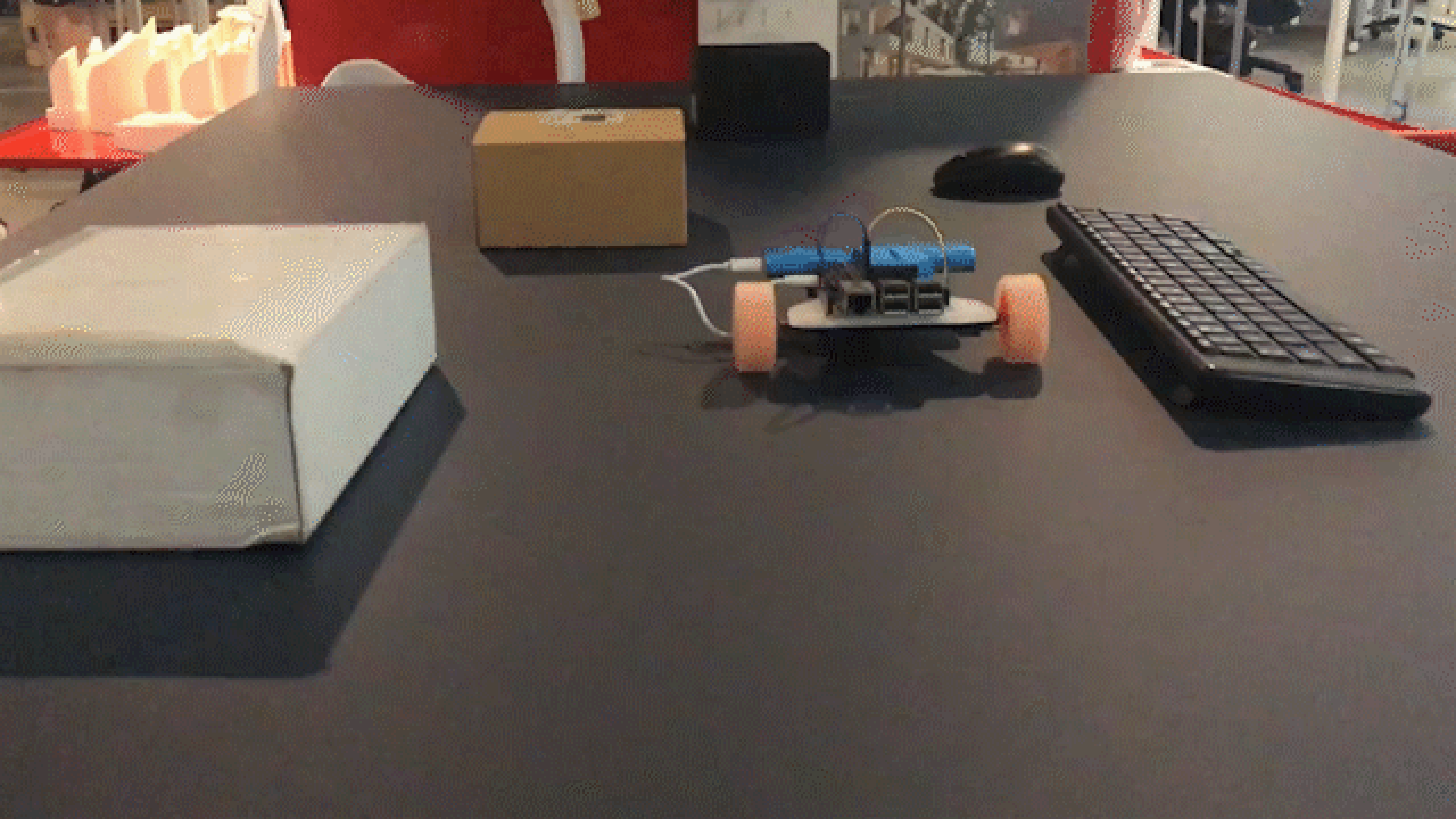
```python
# Main body of code - this detects your key
try:
    while True:
        keyp = readkey()
        if keyp == 'w' or keyp == UP:
            pi2go.forward(speed)
            print 'Forward', speed
        elif keyp == 's' or keyp == DOWN:
            pi2go.reverse(speed)
            print 'Backward', speed
        elif keyp == 'd' or keyp == RIGHT:
            pi2go.spinRight(speed)
            print 'Spin Right', speed
        elif keyp == 'a' or keyp == LEFT:
            pi2go.spinLeft(speed)
            print 'Spin Left', speed

        elif keyp == '.' or keyp == '>':
            speed = min(100, speed+10)
            print 'Speed+', speed
        elif keyp == ',' or keyp == '<':
            speed = max (0, speed-10)
            print 'Speed-', speed
```

Original Script

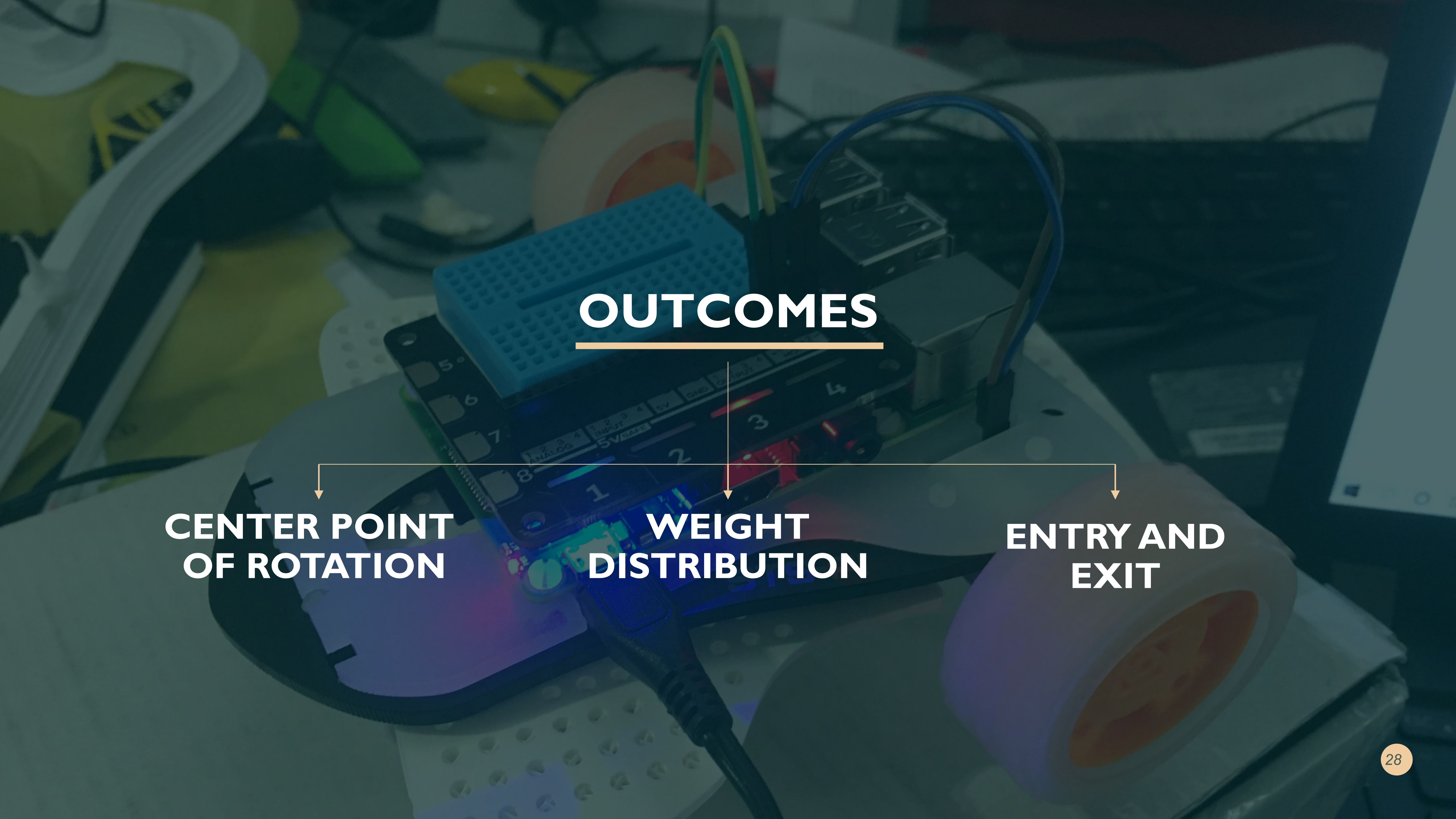


```python
# and changes direction depending on it
try:
    while True:
        keyp = readkey()
        if keyp == 'w' or keyp == UP:
            explorerhat.motor.forward(speed)
            print 'Forward', speed
        elif keyp == 's' or keyp == DOWN:
            explorerhat.motor.backwards(speed)
            print 'Backward', speed
        elif keyp == 'd' or keyp == RIGHT:
            explorerhat.motor.one.backwards(speed)
            explorerhat.motor.two.forwards(speed)
            print 'Spin Right', speed
        elif keyp == 'a' or keyp == LEFT:
            explorerhat.motor.two.backwards(speed)
            explorerhat.motor.one.forwards(speed)
            print 'Spin Left', speed
        elif keyp == '.' or keyp == '>':
            speed = min(100, speed+10)
            explorerhat.motor.forward(speed)
            print 'Speed+', speed
        elif keyp == ',' or keyp == '<':
            speed = max (0, speed-10)
            explorerhat.motor.forward(speed)
            print 'Speed-', speed
```

Fixed Script



OUTCOMES

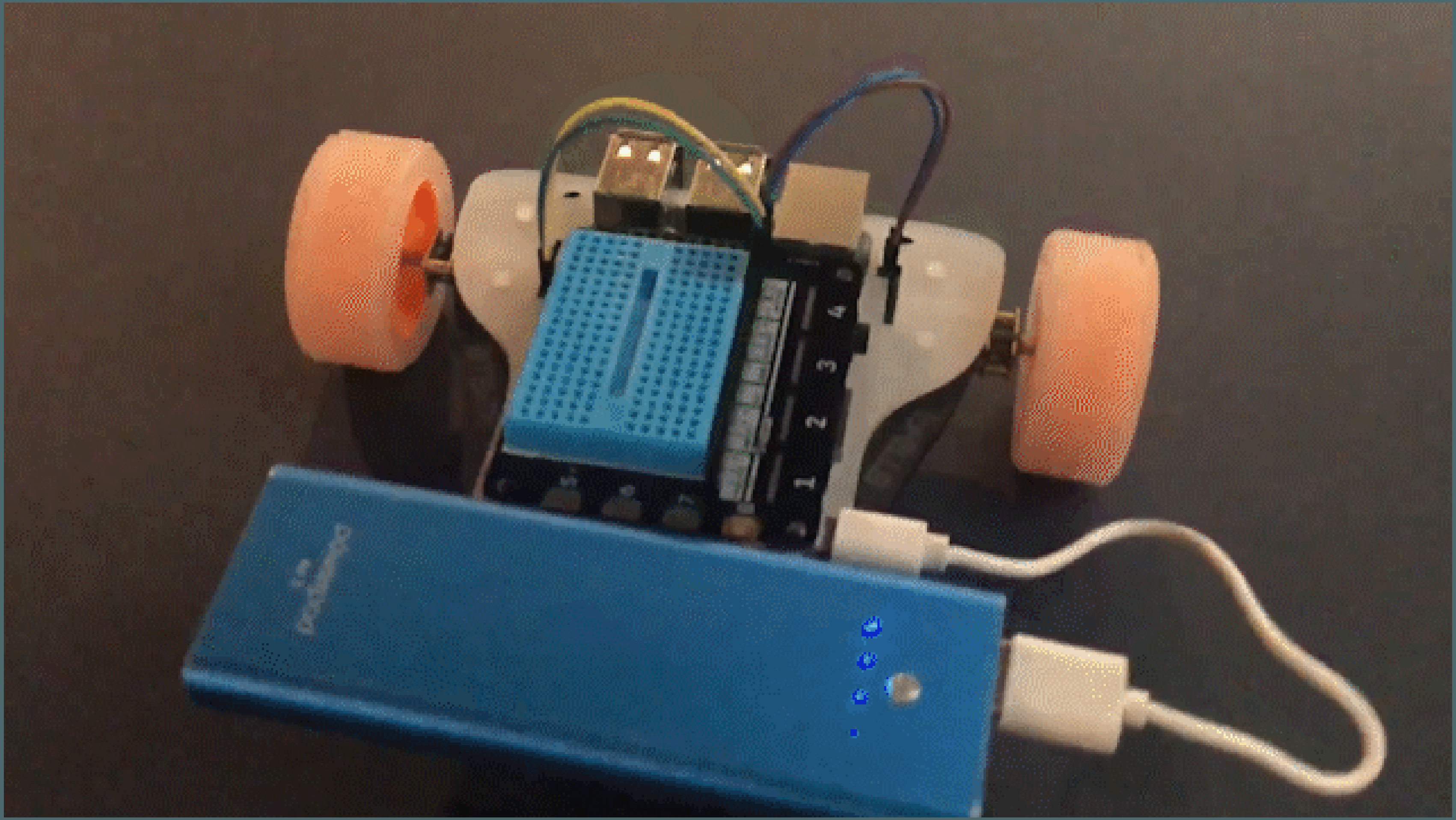


CENTER POINT OF ROTATION

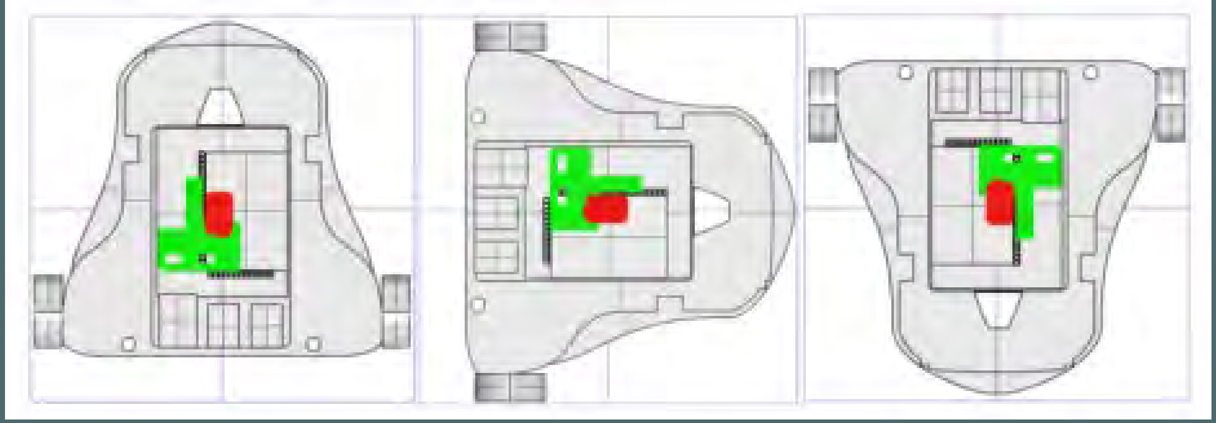
WEIGHT DISTRIBUTION

ENTRY AND EXIT

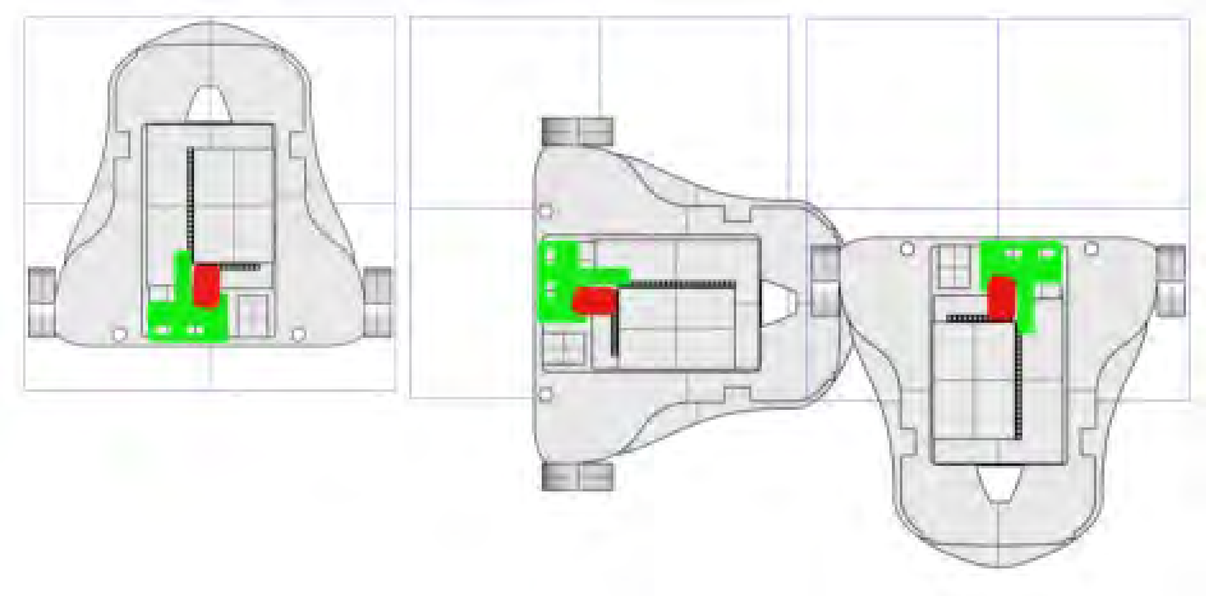
CENTER POINT OF ROTATION



CENTER POINT OF ROTATION

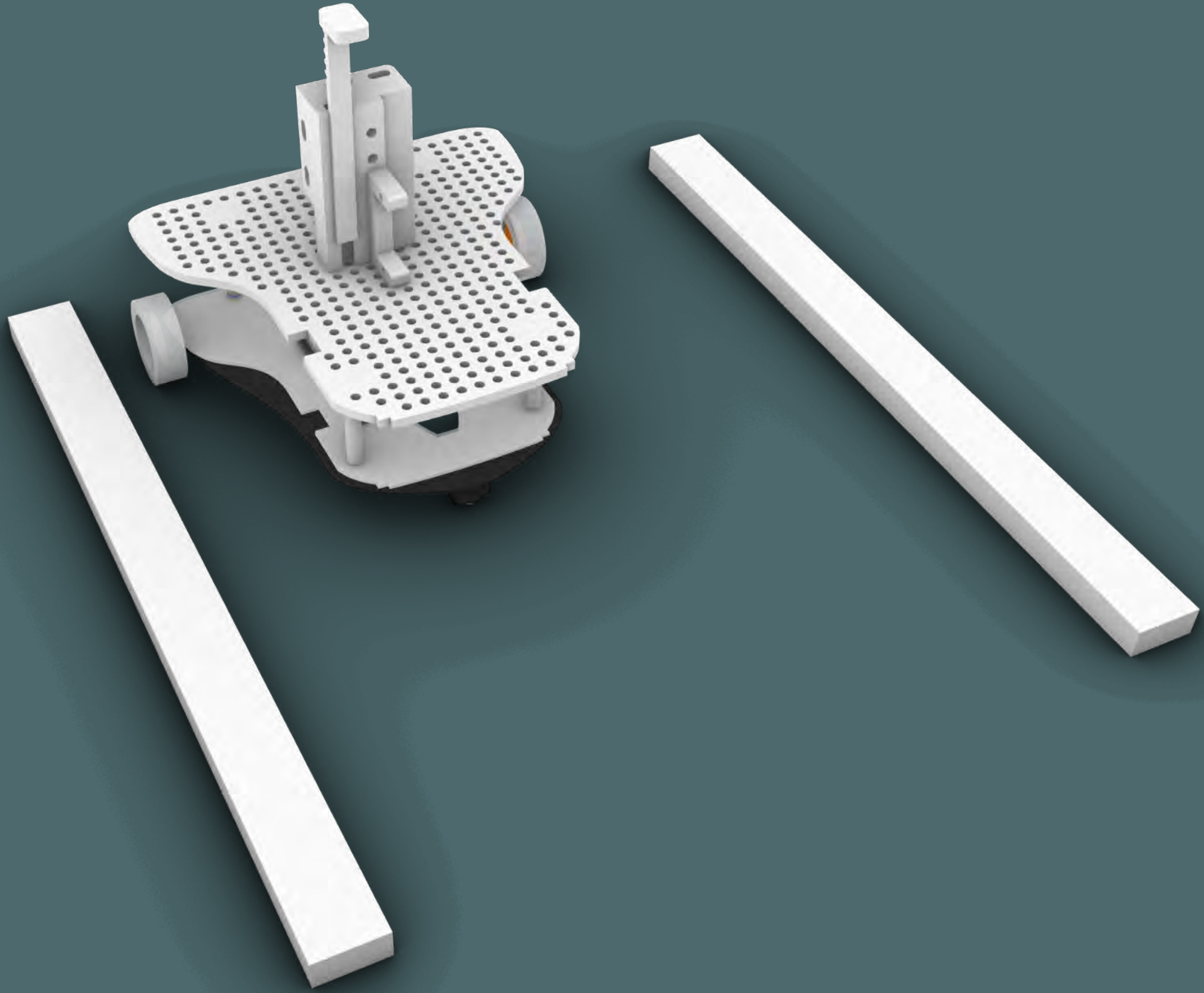


Wanted Rotation

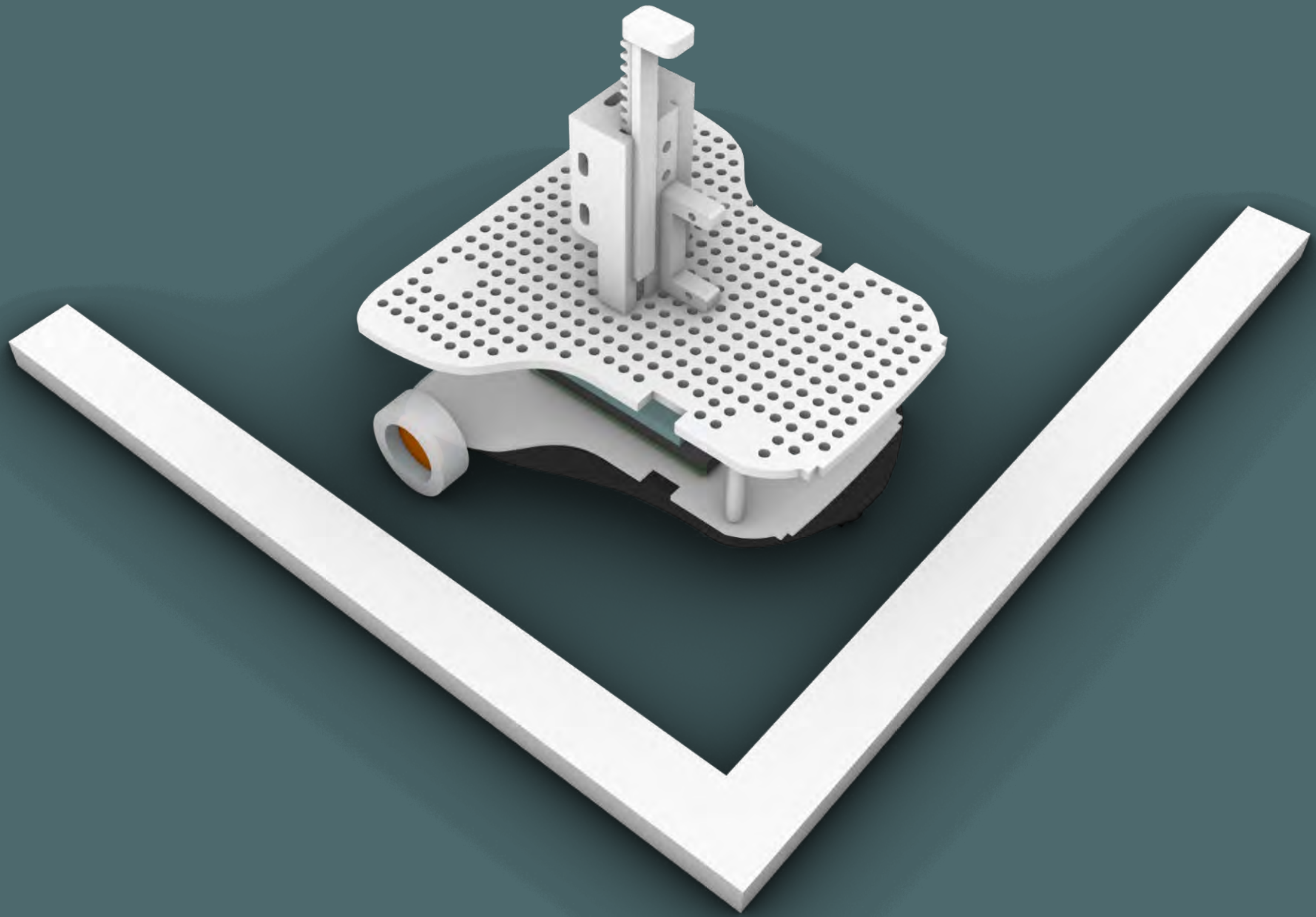


Achieving that Rotation

CENTER POINT OF ROTATION

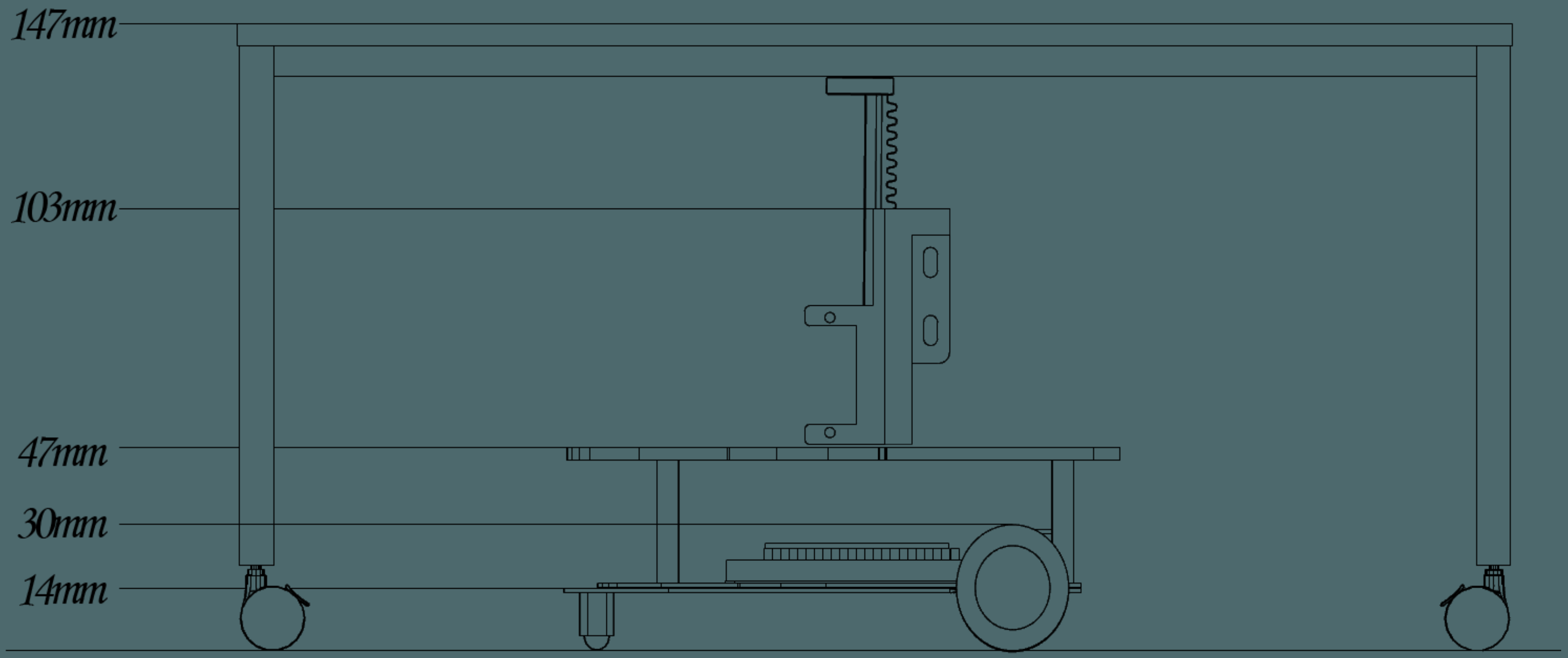


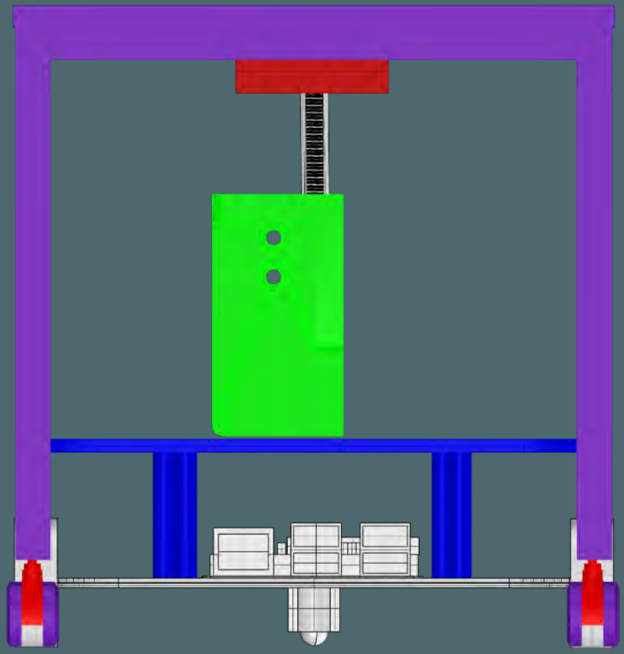
CENTER POINT OF ROTATION



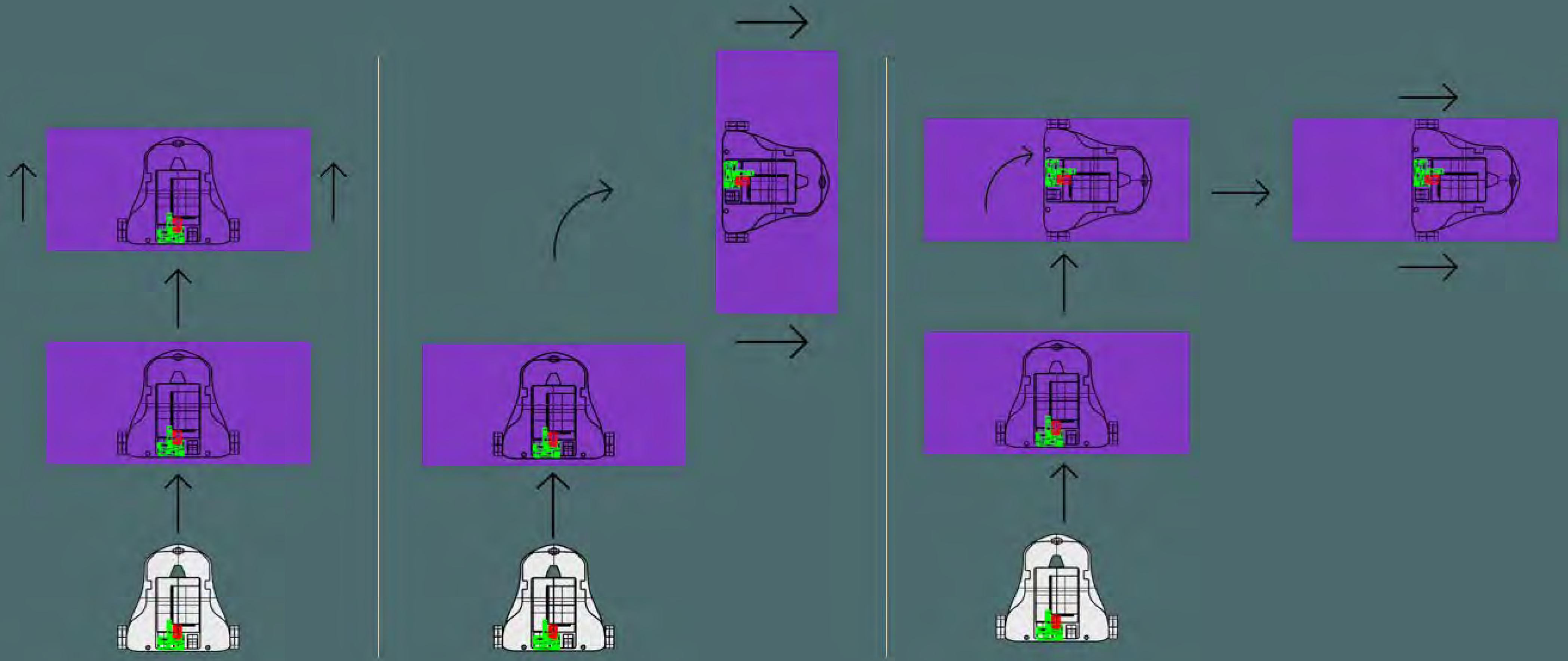
WEIGHT DISTRIBUTION

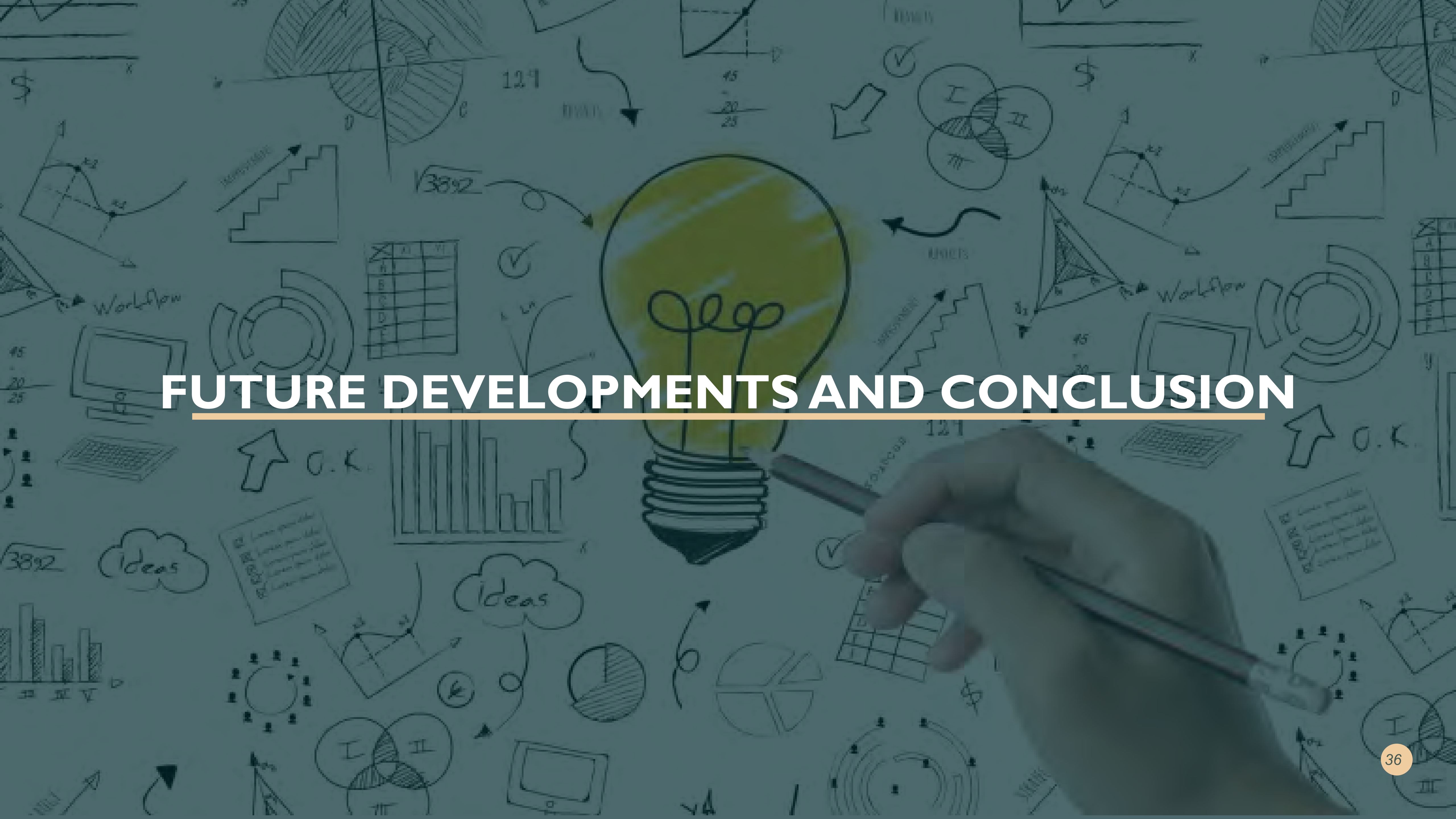
APPOACHING THE DESK (ENTRY AND EXIT)





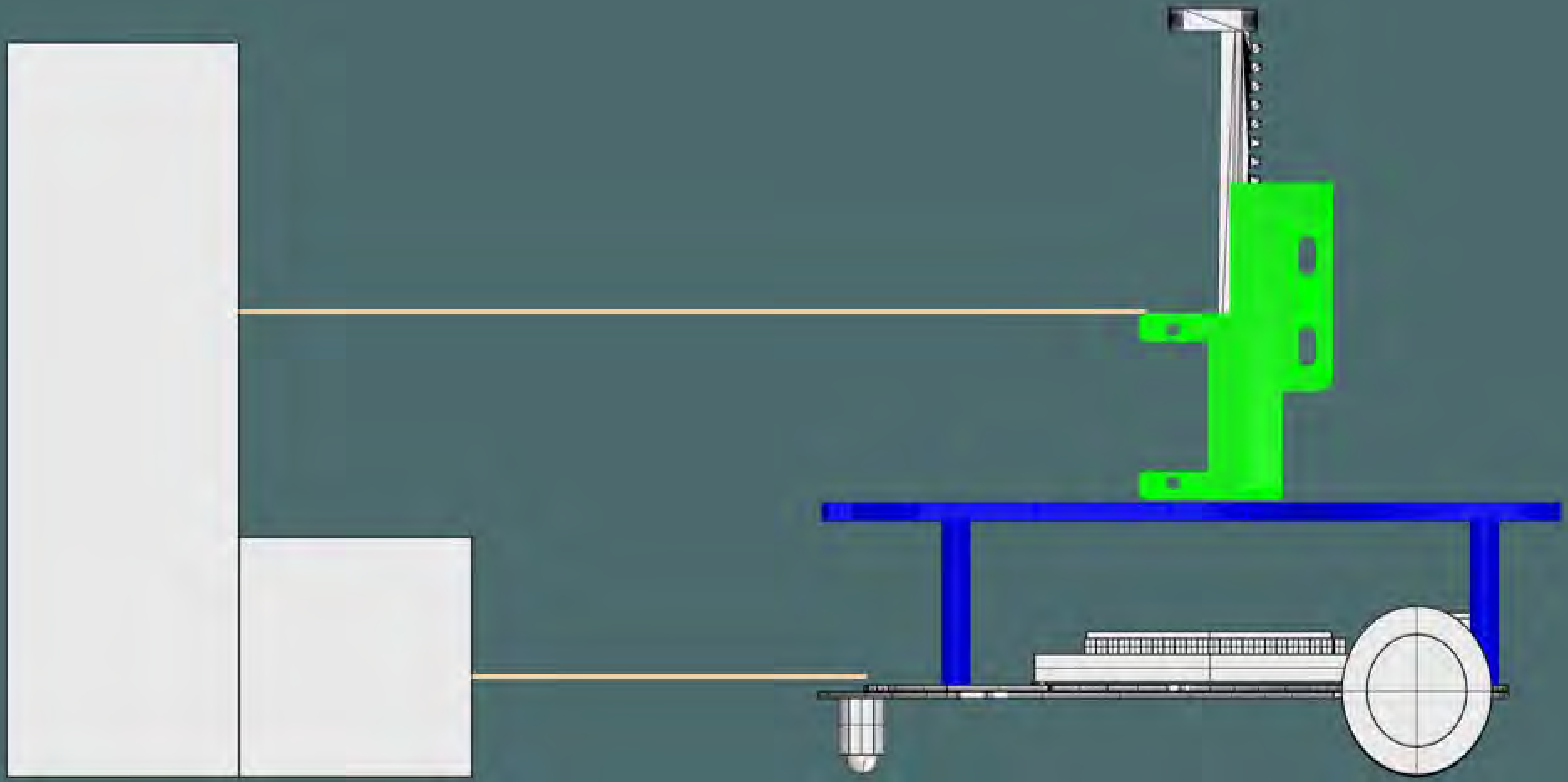
APPOACHING THE DESK (ENTRY AND EXIT)



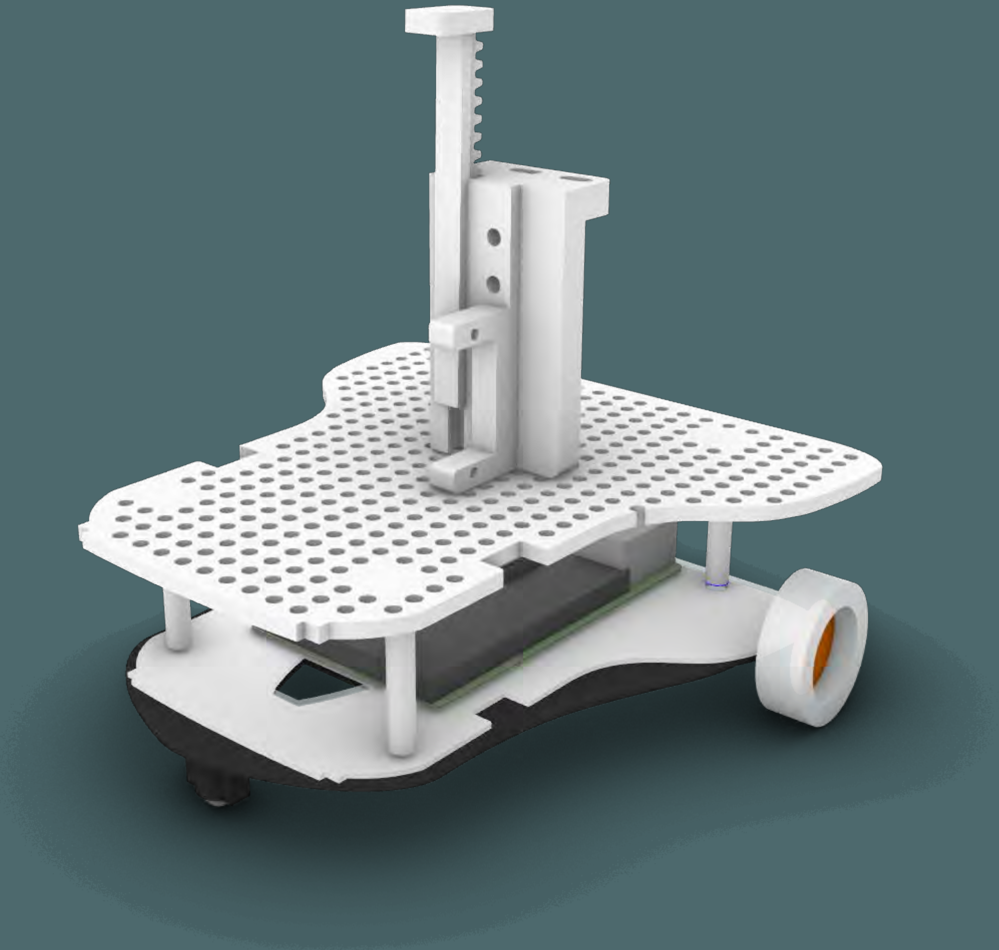


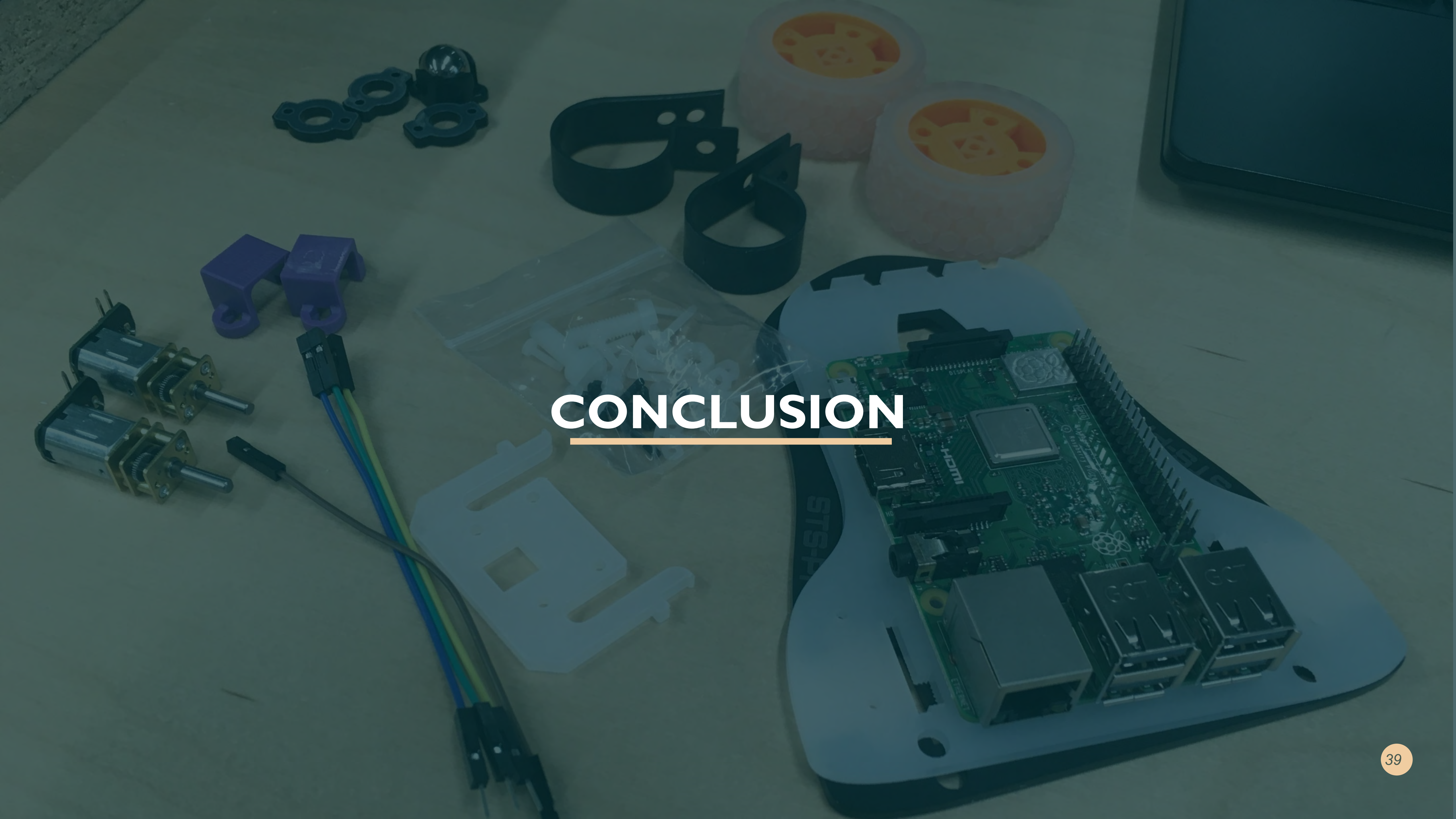
FUTURE DEVELOPMENTS AND CONCLUSION

FUTURE DEVELOPMENTS: AWARENESS



FUTURE DEVELOPMENTS: THE LIFT





CONCLUSION