Design of an Exploration Walking Robot

Electronics

Design Investigation

____

Signature of Sponsoring Teacher

____

Signature of School Science Fair Coordinator

Bianca Verdin
Lane Tech College Prep High School
Chicago, IL 60618
Grade 11

Table of Contents

Acknowledgments

Firstly, I would like to thank all my judges from the Lane Tech, Regional, and City STEM Exhibitions, for providing me with genuine feedback on my project last year. I would also like to thank my parents, Salvador Verdin and Griselda Verdin, for ordering materials and making sure I always had the resources to complete my design. I would also like to apologize to my little brother, Joel Verdin, for the hours of sleep lost due to the 3D printer in his room running all night. Thank you to Divinefavor Osuji, for advising me to upload my code onto a Github repository. Thank you to Mr. Robert Berg, the Lane Tech Robotics Team's coach, for sponsoring my project. And thank you to Mr. Daniel Law, another robotics coach, and my sponsor for last year, for recommending me to Mr. Berg. Go team #4645!

Definition of Need

Robots are now being used for search and rescue missions, mining, worksite inspections, agriculture, and much more. The robots must deal with obstacles, such as steps, doors, or irregular terrain (Bellicoso et al., 2018). As companies such as NASA begin to use walking robots to explore into caves, there is a market for walking robots. According to Xiao and Whittaker (2014), legged robots are less energy-efficient than robots with wheels. This means that the amount of time per mission and the distance that the new walking robots will go will be less than a wheeled rover. Also, many of these cave exploring machines are very large and need long periods of training to control. This is an issue, as it might restrict which caves can be explored. A portable, easy to control and energy-efficient quadruped robot would allow for greater cave exploration and could be easily applied to search and rescue, inspections, and other types of exploration. This design investigation will use the electronic layout and gait from last year's prototype to create a walking robot fit for exploration purposes. The robot uses two microcontrollers. One acts as the Wi-Fi access point and the camera, the other controls the servo motors. Using a computer connected to the Wi-Fi access point, the website controls the robot by sending information to the camera, which then uses the ESP-NOW protocol, which is unique to this brand of microcontrollers, to send the information to the microcontroller that controls the servo motors. The camera also provides feedback to the website via the stream, allowing the user to see what the robot sees.

Background Research

According to Westcott (2020), electronics include controlling the flow of electricity. This subject covers concepts spanning from robotics to electrical engineering. The goal of electrical engineering is to investigate current societal problems and design, build, and test electrical devices that usually improve on previous innovations. Smartphones, computers, and robots are the products of electrical engineering.

One challenge electrical engineers are now facing is designing a robot that can navigate irregular terrain. Most robots use wheels to move around, limiting where the robot can go. However, walking robots instead utilize their legs to move. Their legs provide easy mobility through rough terrain. Currently, Boston Dynamics has created LittleDog and Spot, two quadruped robots designed after dogs. Walking robots can help save lives in disasters and explore risky areas. Researchers and first responders can benefit from using a walking robot. These robots can navigate tight spaces and handle dangerous situations. While they can be very helpful, they are often not practical in their consumption of power. According to Kashiri et al. (2018), Big Dog has a Specific Resistance of .15, so Big Dog can only run for 30 minutes in one full 15-liter tank of fuel. Robots still have a long way to go when it comes to efficiency, especially compared to living things.

Discussion

Walking Robots

According to Wong et al. (2021), “a walking robot is a robot with legged locomotion.” (2) This means that instead of using wheels or tracks to move around, it uses legs controlled by several servo motors to move around. This comes with multiple advantages, which include the ability to move across difficult terrain. Legs are less likely to get stuck in mud, dirt, grass, or any terrain where a wheel digs itself in; legs also do less damage to the ground than tracks or wheels do (Todd, 2013). However, walking robots are very unstable. Teaching a robot to see and overcome obstacles is also difficult. For example, if a walking robot has to climb steps, the robot must see the step and raise its legs to accommodate (Ortiz & Vinjamuri, 2021). The robot will have one static gait because the stability of the robot is more important than the speed of the robot. A gait is the relationship of animal limbs at a certain time and place. It includes the regularly repeating pattern of when the limbs hit the floor and how the animal moves its limbs. Quadruped robots have two groups of gaits, static and dynamic gait. Static gait means that the center of the mass of the robot is always being held up by a triangle. It is very stable, but it is also very slow (Gong et al., 2018). Various gaits make the robot more or less stable. A creeping gait is when the robot only lifts one leg at a time (Kajita & Espianu, 2008). Because it only raises one leg at a time, that triangle always supports the robot but is not the most efficient way to move.

Biped vs. Quadruped Robots

There are multiple types of legged robots, such as bipedal and quadrupedal robots. Biped robots have two legs and are usually humanoid. The center of mass is a lot higher than the center of mass of most quadruped walking robots, which makes them more unstable. However, due to there being fewer legs, depending on the robot, it can be more energy-efficient than a quadruped robot. Quadruped robots use four legs. There are two different quadrupeds: sprawling and mammal-type.(Kitano et al., 2016). Mammals, for example, dogs or cats, biologically inspired the mammal type quadruped. The joints are underneath the body of the robot and their limbs are shoulder-width apart. This type of robot is popular because of its cost-efficiency. Sprawling-type quadrupeds are like spiders, but with fewer legs. The legs come up from the body and then down towards the floor, and the distance between the limbs and the body is wider. This makes the robot’s center of mass lower, which gives the added benefit of the robot being more stable (Kitano et al., 2016). There are other types of walking robots with more legs, such as hexapods, but they are costly and less energy-efficient. Gong et al., (2018) found that quadrupedal robots are more stable and capable of carrying heavier loads compared to one-legged and biped robots.

Current Innovations in Quadruped Robots

Although there is more focus on the research of walking robots, they are not a new concept. Chebyshev invented the first walking robot in the 1800s and the first autonomous quadruped robot was created in 1960, called the “phony pony”, in California (Biswal & Mohanty, 2020). The fastest quadruped robot was the Cheetah by Boston Dynamics in 2013. They made a new mini Cheetah created by MIT in 2021, and it is the first to do a backflip (Zewe, 2021). The robot’s flexibility and speed are useful, but its leaping ability is even more fascinating. Not only that, but it also uses a two-part controller; one controller controls the robot while the other one is a neutral controller that takes in the ground’s view and “learns” from experience. The robot then gets both benefits of blind robots (robots without any cameras or sensors) and improves on robots that would need a map of the area first (Zewe, 2021). The new smallest walking robot is a microscopic insect-like robot that has various gaits and can even jump (Morris, 2022). Its small build can be very helpful in biology, especially in studying small ecosystems and insects. The most interesting thing, however, is not its size, but the way it moves. Instead of using electricity, the tiny robot is made using an alloy which bends when heated. By rapidly and repeatedly heating the robot, they can make the robot walk. The scientist put a glass on the robot to help the alloy return to its shape (Morris, 2022). TITAN XIII, the newest quadruped of the series, is a battery-powered sprawling quadruped robot that can walk efficiently at 1.38 m/s speed (Biswal & Mohanty, 2020). Walking robots have proven their ability to be used in situations such as inspections. The ANYmal could navigate through an oil site and measure different gauges, and the position of valve handles(Bellicoso et al., 2018). These inspections improve the health and safety of the worksite as well as improve efficiency and production.

Energy Efficiency

Specific resistance measures the energy-efficiency of a robot. It is the ratio of the amount of energy in Joules over the gravitational potential energy of the robot, if somebody raised it by the distance the robot walked. (E/Mgd). Another formula is power in watts divided by the mass times the gravitational field strength times velocity. (P/Mgv)This is equivalent to the cost of transport (Kajita & Espianu, 2018). The formula accounts for mass, gravity, distance, and surface friction to calculate the total energy used for movement (Xiao & Whittaker, 2014). There have been numerous designs to lower the energy consumption of walking robots. For example, a robot, made by Wong et al., uses springs up and down the robot's legs to replicate how living things absorb the shock of stepping. However, the robot is still not as efficient as animals or people. The Walkman robot shows a maximum consumption of 368 W, having an SR of 1.35. The SR is seven times higher than a person walking. (Kashmiri et al., 2018.)

Wireless Connection

The ESP32 CAM can connect wirelessly to other devices using either Bluetooth or Wi-Fi. Wireless connection works by sending and receiving data between devices using radio waves (Anniss, 2014, p.5). Although Bluetooth is the common and cheaper option, it lacks range. So instead, the controller uses Wi-Fi to control the robot. (Katona & Kovari, 2016). Wi-Fi is a type of wireless local area network (WLAN) that can cover at least a hundred meters (Banerji & Chowdhury, 2013). A web server is how multiple devices connect. One device acts as an access point (AP), which creates and hosts the WLAN, and other devices that connect to the host are clients. In this case, the ESP32 CAM acts as the access point for the WLAN, and the computer and the ESP WROOM are the clients. The WLAN allows the computer to control the robot via the website, sending the information back to the host, where it is then sent to the microcontroller that controls the servo motors.

Related Studies

The NASA Jet Propulsion Laboratory, the California Institute of Technology, and CoSTAR are currently exploring ways to map and explore Martian caves using autonomous robots. One of these autonomous robots is none other than Boston Dynamics Spot, which is one of the quadruped robots previously mentioned (Bourman et al. 2020). Back in 2020, Bouman et al. used the DARPA Subterranean Challenge, a robotics competition, as a way to test the ability of legged robots that were integrated with NeBULA autonomy, to map areas and complete complex tasks. They even won first place in the DARPA Subterranean Challenge in 2020.

Conclusion

For now, the energy-efficiency of walking robots pales in comparison to wheeled robots or living things. Many designs for a more energy-efficient robot have been made, in hopes that they will be used for research or rescue. There should be more research on using Wi-Fi to control walking robots. Many robots that use a wireless connection to control the robot were wheeled. This design benefited from the research on the cost of transportation. Understanding the meaning behind the cost of transportation and how to measure it will help to compare to previous designs. Research on wireless connection was relevant as the ESP32-CAM acts as the access point for the computer. In the future, more research on self-correcting autonomous robots will have to be made to correct the drift that the robot experiences.

Materials

- 8 Tiankong Rc MG90S micro servo motors with servo horns
- 1 ESP32 WROOM with 38 pins
- 1 ESP32 breakout board
- 1 switch
- 4 1.5v keeppower rechargeable batteries
- 1 3D Printer with PLA Filament
- 22 gauge wire
- 1 laptop
- 1 Micro USB cable
- 1 #0 Phillip screwdriver
- 16 #0-80 x ¾ in. Panhead Phillip screws
- 8 #0-80 x 1/8 in. Panhead Phillip screws
- 1 drill with a Philip drill bit
- 1 battery mount with 4 cells
- 2 25-pin circuit boards
- 1 roll of electrical tape
- 1 roll of masking tape
- 1 scale
- 1 multimeter
- 1 stopwatch

Design Plan

Walking robots are able to deal with unpredictable and irregular terrain. However, to create the perfect robot for exploration or search and rescue, it must obtain certain goals. Firstly, it must be energy-efficient, as the battery must be able to last for long periods of time between charges. Secondly, it must be small and lightweight, allowing it to go into spaces that may be too small for a person. The size and weight will also affect the battery-life of the robot. The robot must be user-friendly and easy to control, for example, wirelessly through the website. Lastly, the robot must have a camera to explore caves or look for survivors and send the video feed to the app in real time.

Table 1.1

Design Criteria	What Makes Up this Design Criteria	Importance on a Scale of 1-5
Functionality of Robot/Website	● The robot is able to walk for at least 10 seconds. ● The robot walks in the direction dictated by the website. ● Camera sends a video feed in real time to the website.	5
Energy-Efficiency	● The robot has a median cost of transportation of around .15 or less ● The cost of transportation is precise.	4

Table 1.1 shows the design criteria of the walking quadruped robot.

Part A: Assembling the Robot

Figure 1.1

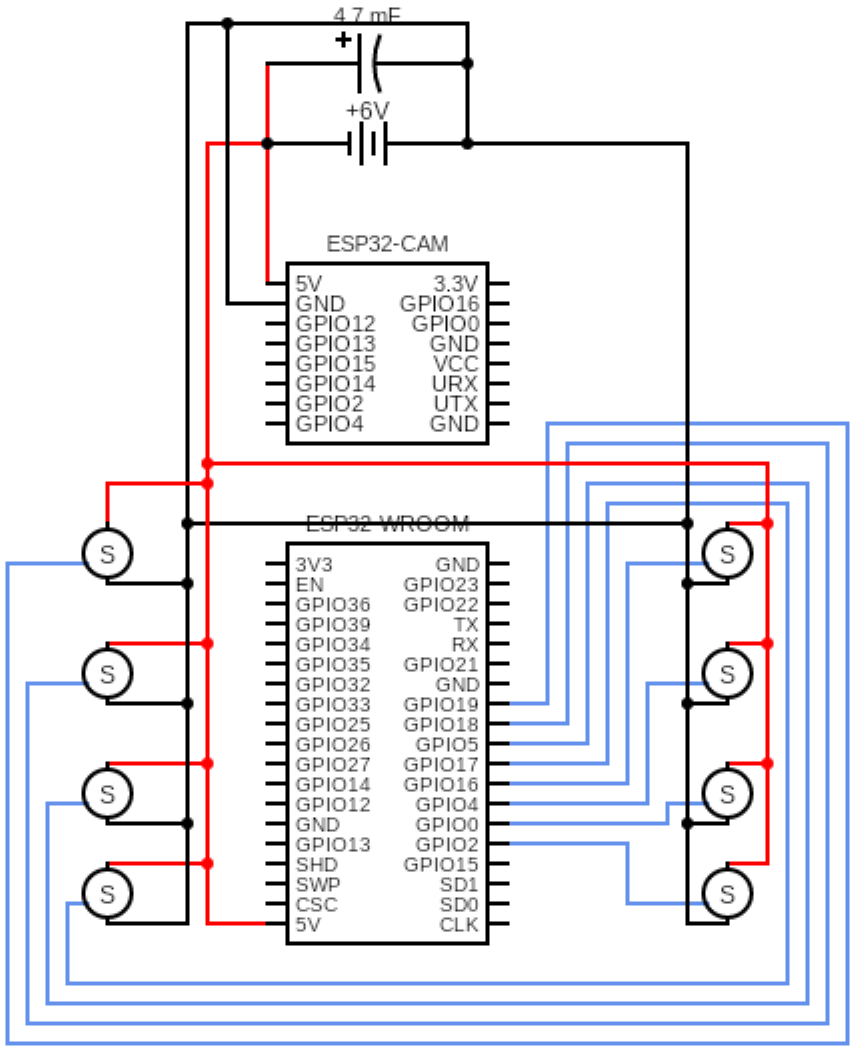

Figure 1.1 is a circuit schematic of the walking robot. Red is power, black is ground, and blue is signal.

1.

Design Results

Table 1.2

Design A

	Distance (m)	**Time (s)**	**Velocity(m/s)**	**CoT**
Trial 1	**0.592**	**419**	**0.001**	**3.44E-07**
Trial 2	**1.365**	**791**	**0.002**	**2.23E-07**
Trial 3	**0.145**	**266**	**0.001**	**2.09E-07**
Trial 4	**0.325**	**213**	**0.002**	**7.31E-07**
Trial 5	**0.396**	**186**	**0.002**	**1.17E-06**
Mean	**0.565**	**375.0**	**0.002**	**5.35E-07**
SD	**0.425**	**223.1**	**0.000**	**3.69E-07**
SE	**0.245**	**128.8**	**0.000**	**2.13E-07**

Table 1.2 shows the distance walked in meters, the velocity, and the cost of transportation across five trials.

Figure 1.3

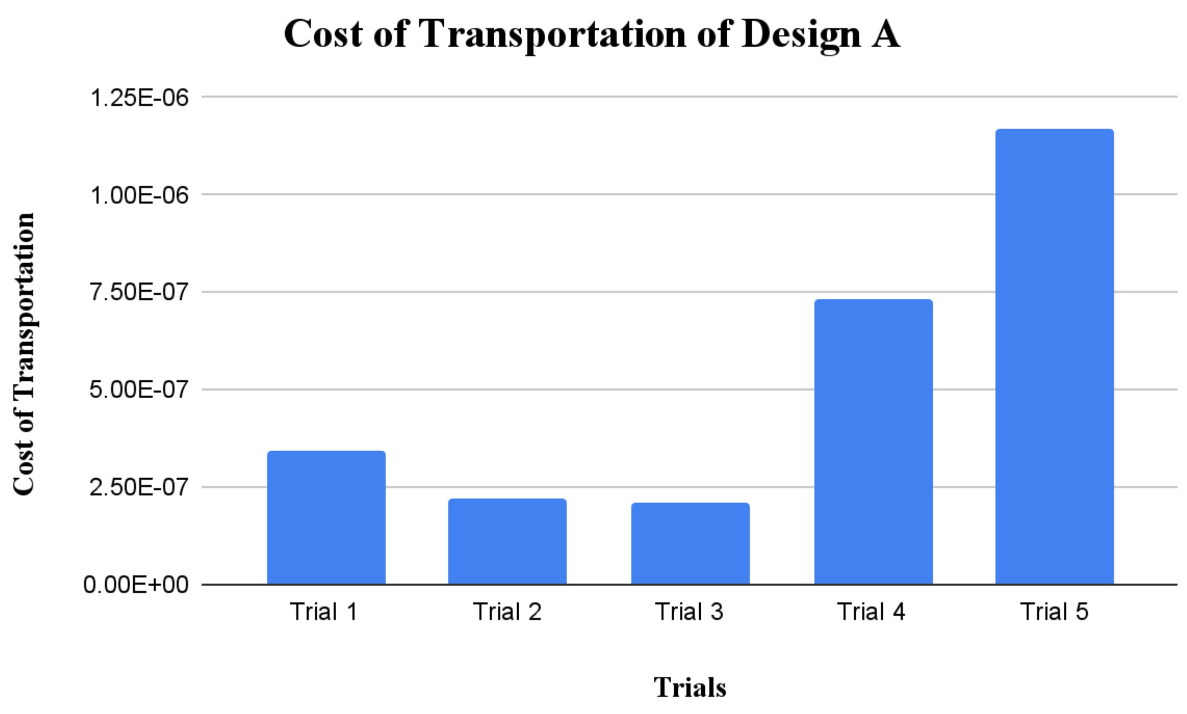

Figure 1.3 shows the cost of transportation of Design A through five trials.

Table 1.3

Design Criteria	What Makes Up this Design Criteria	Completion on a Scale of 1-5
Functionality of Robot/App	● The robot is able to walk for at least 10 seconds. ● The robot walks in the direction dictated by the app. ● Camera sends a video feed in real time to the app.	5

Energy-Efficiency	● The robot has a median cost of transportation of around .15 or less ● The cost of transportation is precise.	5

Table 1.3 shows the completion of the design criteria for Design A.

Redesign Plan

The legs of the robot would slightly splay outwards. A reason could be that there is too much force on the legs, and the heavier, longer body was too much for the stubby legs. When designing the legs, the new design took the measurements from the previous model. However, the body of this prototype is much longer and wider due to there being two microcontrollers inside the robot. Luckily, there is a simple solution. All that needs to be done is to redesign the legs to have more support at the bottom and to be longer.

Part D: Redesign

1. Access the 3D model of the shell of the legs.
   https://www.tinkercad.com/things/6SStwJWXF3I
2. Remove the legs from the servo motors.
3. Attach the new legs onto the servo motors and screw them into place.

Redesign Results

Table 1.4

Design B

	Distance (m)	**Time (s)**	**Velocity(m/s)**	**CoT**
Trial 1	**0.120**	**16**	**0.008**	**4.78E-05**
Trial 2	**0.049**	**13**	**0.004**	**5.58E-05**
Trial 3	**0.095**	**32**	**0.003**	**9.47E-06**
Trial 4	**0.081**	**12**	**0.007**	**5.74E-05**
Trial 5	**0.115**	**29**	**0.004**	**1.40E-05**
Mean	**0.09**	**20.40**	**0.005**	**3.69E-05**
SD	**0.03**	**8.40**	**0.002**	**2.08E-05**
SE	**0.01**	**4.85**	**0.001**	**1.20E-05**

Table 1.4 shows the distance in meters, the velocity, and the cost of transportation across five trials.

Figure 1.4

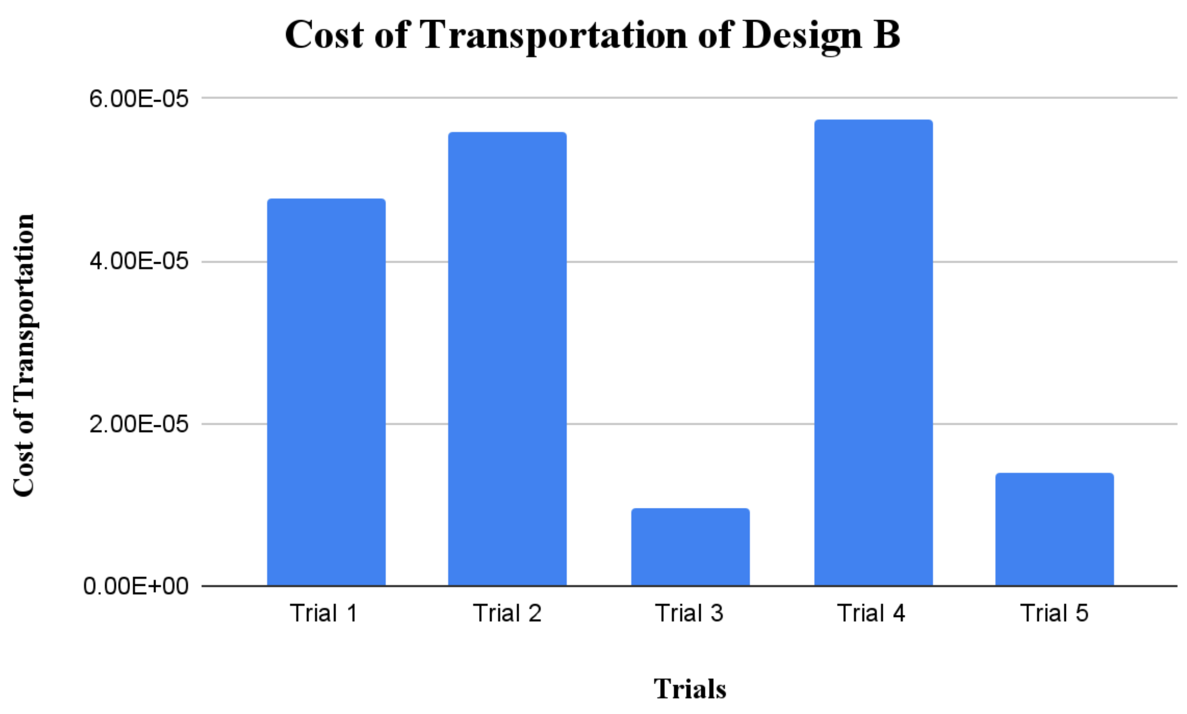

Figure 1.4 shows the cost of transportation of Design B across five trials.

Table 1.5

The Effect of Distance (m) and Time (s) on the Cost of Transport

	Design A				**Design B**			
	Distance (m)	**Time (s)**	**Velocity(m/s)**	**CoT**	**Distance (m)**	**Time (s)**	**Velocity(m/s)**	**CoT**
Trial 1	**0.592**	**419**	**0.001**	**3.44E-07**	**0.120**	**16**	**0.008**	**4.78E-05**
Trial 2	**1.365**	**791**	**0.002**	**2.23E-07**	**0.049**	**13**	**0.004**	**5.58E-05**
Trial 3	**0.145**	**266**	**0.001**	**2.09E-07**	**0.095**	**32**	**0.003**	**9.47E-06**
Trial 4	**0.325**	**213**	**0.002**	**7.31E-07**	**0.081**	**12**	**0.007**	**5.74E-05**
Trial 5	**0.396**	**186**	**0.002**	**1.17E-06**	**0.115**	**29**	**0.004**	**1.40E-05**
Mean	**0.565**	**375.0**	**0.002**	**5.35E-07**	**0.09**	**20.40**	**0.005**	**3.69E-05**
SD	**0.425**	**223.1**	**0.000**	**3.69E-07**	**0.03**	**8.40**	**0.002**	**2.08E-05**
SE	**0.245**	**128.8**	**0.000**	**2.13E-07**	**0.01**	**4.85**	**0.001**	**1.20E-05**

Table 1.5 shows the distance, the velocity, and cost of transportation of design A and design B.

Figure 1.5

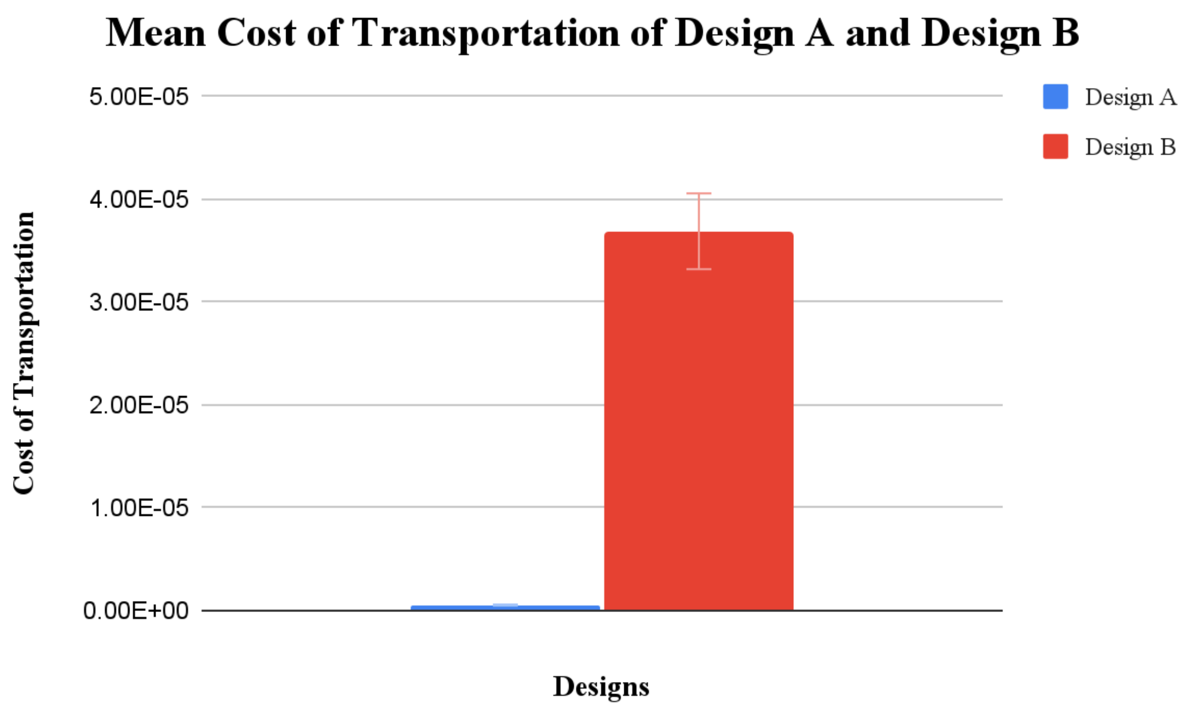

Figure 1.5 shows the mean cost of transportation between the two designs.

Table 1.6

Design Criteria	What Makes Up this Design Criteria	Completion on a Scale of 1-5
Functionality of Robot/App	● The robot is able to walk for at least 10 seconds. ● The robot walks in the direction dictated by the app. ● Camera sends a video feed in real time to the app.	4

Energy-Efficiency	● The robot has a median cost of transportation of around .15 or less ● The cost of transportation is precise.	5

Table 1.6 shows the completion of the design criteria for Design B.

Conclusion

The redesign was in fact worse than the initial design. The motors constantly failed, and the legs were popping off more. The cost of transportation was also worse, as the redesign had a mean Cost of Transportation of 3.69E-5, compared to the initial design having a mean Cost of Transportation of 5.35E-7. I believe that the reason that the robot struggled to walk and had a greater Cost of Transportation when it had the longer legs is due to weight. When designing the legs, the top part was very thick to enclose the wires and the servo motors, which was why the legs were popping off the servo motors. Increasing the length of the legs also meant an increase in mass that the servo motors had to move. That meant that the servo motors had to use more force to move the legs, and when the servo was unable to move the legs, it would lead to all the motors failing. This also explains why the Cost of Transportation was worse in the redesign, because the robot had to do more work, but couldn't walk for even a minute, the power was larger than in the initial design, and while the redesign was heavier than the initial design (0.530kg and 0.500kg respectively), and the velocity of the redesign was greater, due to the larger legs, in the end, it was great enough for it to be as energy-efficient as the initial design.

There are a few issues to note. For example, because the redesign couldn't walk with the protective cover on top, both designs were massed and tested without the protective cover. Another error is that instead of recharging the batteries after each trial, the trials were run consecutively. That means that over time, the motors slowly became weaker. It is most evident in the later trials of the initial design. Another issue was that the robot had a tendency to drift to the left or right, the distance walked might be slightly greater than the recorded distance. Human error should also be taken into account when it comes to the time, as the stopwatch may have not started or stopped at the exact same time the robot begins moving forward or when it experienced a failure.

The purpose of this design was to create an energy-efficient walking quadruped robot that would be able to aid in exploration, but the camera and website could be used for more than just cave exploration. For example, companies could use walking robots with cameras for inspections, as some already have begun doing. The website is easily customizable to fit various tasks. The camera itself has a variety of functions, for example, the camera comes pre-equip with facial detection AI. The camera's compatibility with AI has led to an upcoming design investigation to have the robot run autonomously and use the camera with AI to detect the distance between obstacles.

Next year, during the pursuit of an autonomous walking robot, the top section of the legs will be made thinner to avoid the issue of the legs popping off, as this wasn't an issue that last year's design investigation had, and it had thinner legs. Researching AI will be important for next year's project. Next year's project will develop both the autonomous, but blind, robot from the previous year and the current camera-equipped robot that relies on user input into a robot with the ability to both run by itself and observe its surroundings.

Reference List

Anniss, M. (2014) *How does Wifi work?* Gareth Stevens Publishing. https://www.google.com/books/edition/How_Does_WiFi_Work/FotfEAAAQBAJ?hl=en&gbpv=0

Banerji, S. & Chowdhury, R. S. (2013) On ieee 802.11: Wireless lan technology. International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol. 3, Issue. 4, 2013. 10.5121

Bellicoso, D., Bjelonic, M., Wellhausen, L. & Holtmann, K. (2018) Advances in real-world applications for legged robots. Journal of Field Robotics.10.1002/rob.21839

Biswal, P. & Mohanty P. K. (2020). Development of quadruped walking robots: A review. *Ain Shams Engineering Journal*, 12(2).10.1016

Bouman et al. (2020) Nebula: Team costar's robotic autonomy solution that won phase II of darpa subterranean challenge. "Field robotics", March 2022, vol. 2, p. 1432-1506.10.55417

Gong, D., Wang, P., Zhao, S., Du, L. & Duan, Y. (2018) *Bionic quadruped robot dynamic gait control strategy based on twenty degrees of freedom.* IEEE. 10.1109/JAS.2017.7510790

Kajita, S. & Espiau, B. (2018) Legged Robots. Handbook of robotics. (361-389) Springer. 978-3-319-32552-1

Katona, J. & Kovari, A. (2016). Cost-effective wifi controlled mobile robot. 11th International Symposium on Applied Informatics and Related Areas. https://www.researchgate.net/profile/Jozsef-Katona/publication/310600119_Cost-effective_WiFi_controlled_mobile_robot/links/5b3a5423a6fdcc8506ea375f/Cost-effective-WiFi-controlled-mobile-robot.pdf

Kashiri et al. (2018) An Overview on Principles for Energy Efficient Robot Locomotion. Advances in Mechatronics and Biomechanics towards Efficient Robot Actuation. https://doi.org/10.3389/frobt.2018.00129

Kitano, S., Hirose, S., Horigome, A. & Endo, G. (2016) Titan-XIII: Sprawling-type quadruped robot with ability of fast and energy-efficient walking. ROBOMECH Journal, 3(8). https://doi.org/10.1186/s40648-016-0047-1

Morris, A. (2022) Tiny robotic crab is smallest-ever remote-controlled walking robot. Northwestern. https://news.northwestern.edu/stories/2022/05/tiny-robotic-crab-is-smallest-ever-remote-controlled-walking-robot/

Ortiz, J. H. & Vinjamuri R. (2021) Collaborative and humanoid robots. Intech Open. https://www.google.com/books/edition/_/GtJGEAAAQBAJ?hl=en&gbpv=0

Todd, D. J. (2013). *Walking machines*. Springer US.https://www.google.com/books/edition/Walking_Machines/zb_SBwAAQBAJ?hl=en&gbpv=0

Westcott, S., & Westcott J. R. (2020) Basic electronics: Theory and practice. Mercury Learning and Information.https://www.google.com/books/edition/Basic_Electronics/mkbrDwAAQBAJ?hl=en&gbpv=0

Wong, L. H., Sivanesan, S., M F A Faisol, Othman, W., Wahab, A. & Alhady, S. (2021) Development of quadruped walking robot with passive compliance legs using XL4005 buck converter. Journal of Physics: Conference Series, 1969(), 26-27. https://doi.org/10.1088/1742-6596/1969/1/012003

Xiao, X. & Whittaker, W. L. (2014) Energy considerations for wheeled mobile robots operating on a single battery discharge. Carnegie Mellon University. https://www.ri.cmu.edu/pub_files/2014/8/TR-14-16.pdf

Zewe, A. (2021). One giant leap for the mini cheetah: A new control system, demonstrated using MIT's robotic mini cheetah, enables four-legged robots to jump across uneven terrain in real-time. MIT News. https://news.mit.edu/2021/one-giant-leap-mini-cheetah-1020

Appendix

ESP32-CAM Complete code:

```
//All my code was uploaded to my repository at
//https://github.com/bsverdin/Walking-Robot.
//Libraries
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_wifi.h"
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"             // disable brownout problems
#include "soc/rtc_cntl_reg.h"    // disable brownout problems
#include "esp_http_server.h"
#include <esp_now.h>
int int_value = 0;
// WiFi Password
const char *ssid1 = "VinnieWiFi";
const char *password1 = "Qwerty123!";
//ESP32-WROOM MAC ADDRESS
uint8_t broadcastAddress[] = { 0xE0, 0x5A, 0x1B, 0xAC, 0x5A, 0xF8 };
#define PART_BOUNDARY "123456789000000000000987654321"
#define CAMERA_MODEL_AI_THINKER
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27
  #define LED_BUILTIN       4
  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22
//ESP NOW
typedef struct struct_message {
  int b;
} struct_message;
struct_message myData;
esp_now_peer_info_t peerInfo;
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" :
"Delivery Fail");
}
static const char* _STREAM_CONTENT_TYPE =
"multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type:
image/jpeg\r\nContent-Length: %u\r\n\r\n";

httpd_handle_t camera_httpd = NULL;
httpd_handle_t stream_httpd = NULL;

static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<html>
  <head>
    <title>Vinnie</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      body { font-family: Arial; text-align: center; margin:0px auto;
padding-top: 30px;}
      table { margin-left: auto; margin-right: auto; }
      td { padding: 8 px; }
      .button {
        background-color: #8563e7;
        border: none;
        color: white;
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 18px;
        margin: 6px 3px;
        cursor: pointer;
        -webkit-touch-callout: none;
        -webkit-user-select: none;
        -khtml-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
        user-select: none;
        -webkit-tap-highlight-color: rgba(0,0,0,0);
      }
      img {  width: auto ;
        max-width: 100% ;
        height: auto ;
      }
    </style>
  </head>
  <body>
    <h1>Walking Exploration Robot</h1>
    <img src="" id="photo" >
    <table>
      <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('forward');"
ontouchstart="toggleCheckbox('forward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Forward</button></td></tr>
      <tr><td align="center"><button class="button"
onmousedown="toggleCheckbox('left');"
ontouchstart="toggleCheckbox('left');" onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Left</button></td><td
align="center"><button class="button"
onmousedown="toggleCheckbox('stop');"
ontouchstart="toggleCheckbox('stop');">Stop</button></td><td
align="center"><button class="button"
onmousedown="toggleCheckbox('right');"
ontouchstart="toggleCheckbox('right');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Right</button></td></tr>
```

```
      <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('backward');"
ontouchstart="toggleCheckbox('backward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Backward</button></td></tr>
      <tr><td></td><td align="center"><button class="button button4"
id="flash" onmousedown="toggleCheckbox('flash');"
ontouchstart="toggleCheckbox('flash');"
onmouseup="toggleCheckbox('flashoff');"
ontouchend="toggleCheckbox('flashoff');">Flash</button></td><td></td></tr>
    </table>
   <script>
   function toggleCheckbox(x) {
     var xhr = new XMLHttpRequest();
     xhr.open("GET", "/action?go=" + x, true);
     xhr.send();
   }
   window.onload = document.getElementById("photo").src =
window.location.href.slice(0, -1) + ":81/stream";
  </script>
  </body>
</html>
)rawliteral";

static esp_err_t index_handler(httpd_req_t *req){
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML,
strlen(INDEX_HTML));
}

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf,
&_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART,
_jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf,
_jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
```

```
        _jpg_buf = NULL;
      } else if(_jpg_buf){
        free(_jpg_buf);
        _jpg_buf = NULL;
      }
      if(res != ESP_OK){
        break;
      }
      //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
    }
    return res;
  }

  static esp_err_t cmd_handler(httpd_req_t *req){
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};

    buf_len = httpd_req_get_url_query_len(req) + 1;
    if (buf_len > 1) {
      buf = (char*)malloc(buf_len);
      if(!buf){
        httpd_resp_send_500(req);
        return ESP_FAIL;
      }
      if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
        if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) ==
  ESP_OK) {
        } else {
          free(buf);
          httpd_resp_send_404(req);
          return ESP_FAIL;
        }
      } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
      }
      free(buf);
    } else {
```

```
    httpd_resp_send_404(req);
    return ESP_FAIL;
  }

  sensor_t * s = esp_camera_sensor_get();
  int res = 0;


   if (!strcmp(variable, "flash")) {
    Serial.println("Flash On");
    pinMode(LED_BUILTIN, HIGH);

  } else if (!strcmp(variable, "flashoff")) {
    Serial.println("Flash Off");
    pinMode(LED_BUILTIN, LOW);
  }
   if(!strcmp(variable, "forward")) {
    Serial.println("Forward");
    int_value = 1;
    // Set values to send
    myData.b = int_value;
    // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  } else if (!strcmp(variable, "left")) {
    Serial.println("Left");
    int_value = 2;
    // Set values to send
    myData.b = int_value;
    // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));
```

```
  if (result == ESP_OK) {
    Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  } else if (!strcmp(variable, "right")) {
    Serial.println("Right");
    int_value = 3;
    // Set values to send
    myData.b = int_value;
    // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  } else if (!strcmp(variable, "backward")) {
    Serial.println("Backward");
    int_value = 4;
    // Set values to send
    myData.b = int_value;
    // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  } else if (!strcmp(variable, "stop")) {
    Serial.println("Stop");
    int_value = 5;
    // Set values to send
    myData.b = int_value;
    // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  } else {
    res = -1;
  }

  if(res){
    return httpd_resp_send_500(req);
  }

  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.softAP(ssid1, password1);
pinMode (LED_BUILTIN, OUTPUT);
  // Init ESP-NOW
  esp_now_init();
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 1;
  peerInfo.encrypt = false;
```

```
// Add peer
if (esp_now_add_peer(&peerInfo) != ESP_OK) {
  Serial.println("Failed to add peer");
  return;
}

camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

// Camera init
esp_err_t err = esp_camera_init(&config);
```

```
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.softAP(ssid1, password1);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  // Start streaming web server
  startCameraServer();
}

void loop() {

}
```

ESP32-WROOM Complete code:

```
//All my code was uploaded to my repository at
//https://github.com/bsverdin/Walking-Robot.
//Libraries
#include <esp_now.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include "esp_wifi.h"
//Chanel
#define CHANNEL 1
//Servos
Servo shoulder1;
Servo elbow1;
Servo shoulder2;
Servo elbow2;
Servo hip3;
Servo knee3;
Servo hip4;
Servo knee4;
//Starting position based on how servos are installed
int s1 = 90;
int s2 = 90;
int h3 = 90;
int h4 = 90;
int e1 = 90;
int e2 = 90;
int k3 = 90;
int k4 = 90;
//Delays
int d1 = 500;
int d2 = 1000;
int d3 = 50;
int d4 = 75;
int d5 = 100;
int d6 = 3000;
int m = 0;
//servo GPIO pins
int s1Pin = 19;
int e1Pin = 18;
int s2Pin = 5;
int e2Pin = 17;
int h3Pin = 16;
int k3Pin = 4;
int h4Pin = 2;
int k4Pin = 15;
//Methods
void walk_fwd() {
  for (m = 0; m <= 12; m += 1) {
    knee3.write(k3 - 60 - m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    hip3.write(h3 + 30 - m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
    knee3.write(k3 - 72 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    hip3.write(h3 + m);
    delay(15);
```

```
}
for (m = 0; m <= 12; m += 1) {
  knee3.write(k3 - 48 - m);
  delay(15);
}
  for (m = 0; m <= 12; m += 1) {
  elbow2.write(e2 + 60 + m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  shoulder2.write(s2 - 30 + m);
  delay(15);
}
for (m = 0; m <= 24; m += 1) {
  elbow2.write(e2 + 72 - m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  shoulder2.write(s2 - m);
  delay(15);
}
for (m = 0; m <= 12; m += 1) {
  elbow2.write(e2 + 48 + m);
  delay(15);
}
for (m = 0; m <= 12; m += 1) {
  knee4.write(k4 + 60 + m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  hip4.write(h4 - 30 + m);
  delay(15);
}
for (m = 0; m <= 24; m += 1) {
  knee4.write(k4 + 72 - m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  hip4.write(h4 - m);
  delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    knee4.write(k4 + 48 + m);
    delay(15);
  }
    for (m = 0; m <= 12; m += 1) {
    elbow1.write(e1 - 60 - m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 + 30 - m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
    elbow1.write(e1 - 72 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 + m);
    delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    elbow1.write(e1 - 48 - m);
    delay(15);
  }
}
void walk_left() {
  for (m = 0; m <= 12; m += 1) {
    knee3.write(k3 - 40 - m);
    delay(0);
    elbow2.write(e2 + 80 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder2.write(s2 - 50 + m);
    delay(0);
    hip3.write(h3 + 10 - m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
```

```
  knee3.write(k3 - 52 + m);
  delay(0);
  elbow2.write(e2 + 92 - m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  shoulder2.write(s2 - m);
  delay(0);
  hip3.write(h3 + m);
  delay(15);
}
for (m = 0; m <= 12; m += 1) {
  knee3.write(k3 - 28 - m);
  delay(0);
  elbow2.write(e2 + 68 + m);
  delay(15);
}
for (m = 0; m <= 12; m += 1) {
  knee4.write(k4 + 80 + m);
  delay(0);
  elbow1.write(e1 - 40 - m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  shoulder1.write(s1 + 10 - m);
  delay(0);
  hip4.write(h4 - 50 + m);
  delay(15);
}
for (m = 0; m <= 24; m += 1) {
  knee4.write(k4 + 92 - m);
  delay(0);
  elbow1.write(e1 - 52 + m);
  delay(15);
}
for (m = 0; m <= 30; m += 1) {
  shoulder1.write(s1 + m);
  delay(0);
  hip4.write(h4 - m);
  delay(15);
```

```
    }
    for (m = 0; m <= 12; m += 1) {
      knee4.write(k3 + 68 + m);
      delay(0);
      elbow1.write(e2 - 28 - m);
      delay(15);
    }
  }
  void walk_right() {
    for (m = 0; m <= 12; m += 1) {
      knee3.write(k3 - 80 - m);
      delay(0);
      elbow2.write(e2 + 40 + m);
      delay(15);
    }
    for (m = 0; m <= 30; m += 1) {
      shoulder2.write(s2 - 10 + m);
      delay(0);
      hip3.write(h3 + 50 - m);
      delay(15);
    }
    for (m = 0; m <= 24; m += 1) {
      knee3.write(k3 - 92 + m);
      delay(0);
      elbow2.write(e2 + 52 - m);
      delay(15);
    }
    for (m = 0; m <= 30; m += 1) {
      shoulder2.write(s2 - m);
      delay(0);
      hip3.write(h3 + m);
      delay(15);
    }
    for (m = 0; m <= 12; m += 1) {
      knee3.write(k3 - 68 - m);
      delay(0);
      elbow2.write(e2 + 28 + m);
      delay(15);
    }
    for (m = 0; m <= 12; m += 1) {
```

```
    knee4.write(k4 + 40 + m);
    delay(0);
    elbow1.write(e1 - 80 - m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 + 50 - m);
    delay(0);
    hip4.write(h4 - 10 + m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
    knee4.write(k4 + 52 - m);
    delay(0);
    elbow1.write(e1 - 92 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 + m);
    delay(0);
    hip4.write(h4 - m);
    delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    knee4.write(k3 + 28 + m);
    delay(0);
    elbow1.write(e2 - 68 - m);
    delay(15);
  }
}
void walk_back() {
  for (m = 0; m <= 12; m += 1) {
    knee4.write(k3 - 48 - m);
    delay(0);
    elbow1.write(e2 + 48 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 - m);
    delay(0);
    hip4.write(h4 + m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
    knee4.write(k4 - 72 + m);
    delay(0);
    elbow1.write(e1 + 72 - m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder1.write(s1 - 30 + m);
    delay(0);
    hip4.write(h4 + 30 - m);
    delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    knee4.write(k4 - 60 - m);
    delay(0);
    elbow1.write(e1 + 60 + m);
    delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    knee3.write(k3 + 48 + m);
    delay(0);
    elbow2.write(e2 - 48 - m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder2.write(s2 + m);
    delay(0);
    hip3.write(h3 - m);
    delay(15);
  }
  for (m = 0; m <= 24; m += 1) {
    knee3.write(k3 + 72 - m);
    delay(0);
    elbow2.write(e2 - 72 + m);
    delay(15);
  }
  for (m = 0; m <= 30; m += 1) {
    shoulder2.write(s2 + 30 - m);
    delay(0);
    hip3.write(h3 - 30 + m);
    delay(15);
  }
  for (m = 0; m <= 12; m += 1) {
    knee3.write(k3 + 60 + m);
    delay(0);
    elbow2.write(e2 - 60 - m);
    delay(15);
  }
}
void walk_stop() {
  elbow2.write(e2 + 60);
  delay(d5);
  shoulder2.write(s2 - 30);
  delay(d5);
  knee4.write(k4 + 60);
  delay(d5);
  hip4.write(h4 - 30);
  delay(d5);
  elbow1.write(e1 - 60);
  delay(d5);
  shoulder1.write(s1 + 30);
  delay(d5);
  knee3.write(k3 - 60);
  delay(d5);
  hip3.write(h3 + 30);
  delay(d6);
}
// WiFi Password
const char* ssid1 = "VinnieWiFi";
const char* password1 = "Qwerty123!";
// Receive data
typedef struct struct_message {
  int b;
} struct_message;
struct_message myData;
// callback function that will be executed when data is received
```

```
void OnDataRecv(const uint8_t* mac, const uint8_t* incomingData, int len)
{
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.println("Data Recieved");
  if (myData.b == 1) {
    Serial.println("Forward");
    while (myData.b == 1){
      walk_fwd();}
   }else if (myData.b == 2) {
    Serial.println("Left");
    while (myData.b == 2){
      walk_left();}
  } else if (myData.b == 3) {
    Serial.println("Right");
    while (myData.b == 3){
      walk_right();}
  } else if (myData.b == 4) {
    Serial.println("Backward");
    while (myData.b == 4){
      walk_back();}
  } else
    Serial.println("Stop");
    walk_stop();
  }

void setup() {
  Serial.begin(115200);
  // Wi-Fi connection
  WiFi.mode(WIFI_AP_STA);
  WiFi.softAP(ssid1, password1, CHANNEL, 1);
  IPAddress myIP = WiFi.softAPIP();
  WiFi.begin(ssid1, password1);
  Serial.println("Connected");
  // Init ESP-NOW
  esp_now_init();
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
```

Unset
```
esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
```

```
hip3.attach(h3Pin);  //purple
hip3.write(h3);
delay(d1);
shoulder1.attach(s1Pin);  //orange
shoulder1.write(s1);
delay(d1);
hip4.attach(h4Pin);  //grey
hip4.write(h4);
delay(d1);
shoulder2.attach(s2Pin);  //green
shoulder2.write(s2);
delay(d1);
knee3.attach(k3Pin);  //black
knee3.write(k3);
delay(d1);
elbow1.attach(e1Pin);  //yellow
elbow1.write(e1);
delay(d1);
knee4.attach(k4Pin);  //white
knee4.write(k4);
delay(d1);
elbow2.attach(e2Pin);  //blue
elbow2.write(e2);
delay(d5);
elbow2.write(e2 + 60);
delay(d5);
shoulder2.write(s2 - 30);
delay(d5);
knee4.write(k4 + 60);
delay(d5);
hip4.write(h4 - 30);
delay(d5);
elbow1.write(e1 - 60);
delay(d5);
```

```
  shoulder1.write(s1 + 30);
  delay(d5);
  knee3.write(k3 - 60);
  delay(d5);
  hip3.write(h3 + 30);
  delay(d5);
}
void loop() {
}
```