6) Servo motor

One micro servo motor is used to slightly turn the ultrasonic sensor from right to left when the robot detects an obstacle in front of itself. This component acts as the mounting point for the ultrasonic sensor that continuously runs the path avoidance code throughout both levels of the robot. Its connection to the Arduino is seen in Fig. 12.6.

7) Solar power

ReGenBot uses solar panels to provide the necessary power for all subsystems. A lithium-ion rechargeable battery that provides a 5V source from a single cell is used. (Fig. 13)

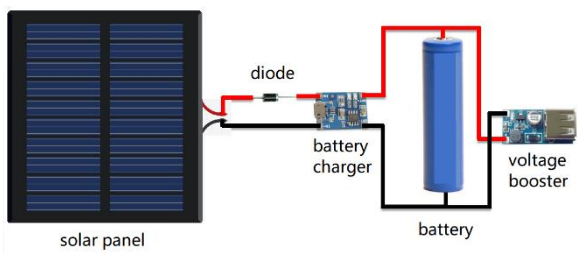

Fig. 13 Circuit for solar panel

C. Coding

ReGenBot's code is developed on Arduino's IDE software. The development and execution of this program are divided into six parts: terrain and program level input, sensor data collection, obstacle avoidance, path correction, and navigation. The integration of these subsystems is displayed in the flowchart in Fig. 14 and Fig. 15 and discussed in further detail in the following subsections.

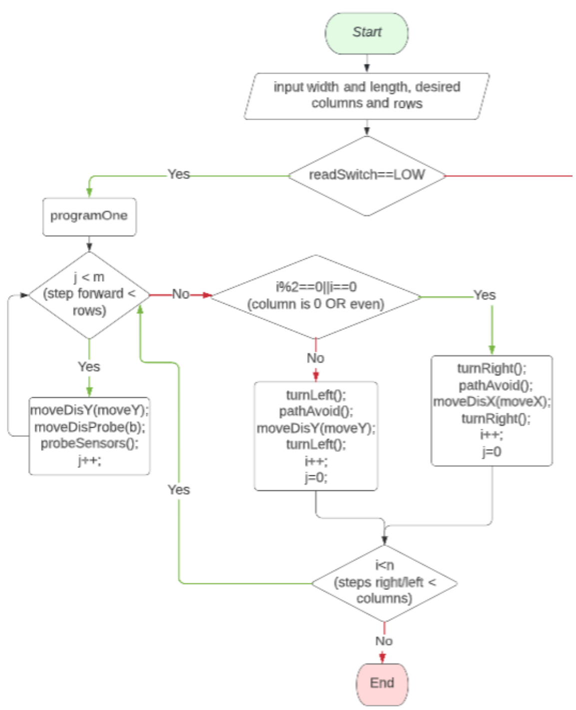

Fig. 14 Overview of the system input and programOne flowchart

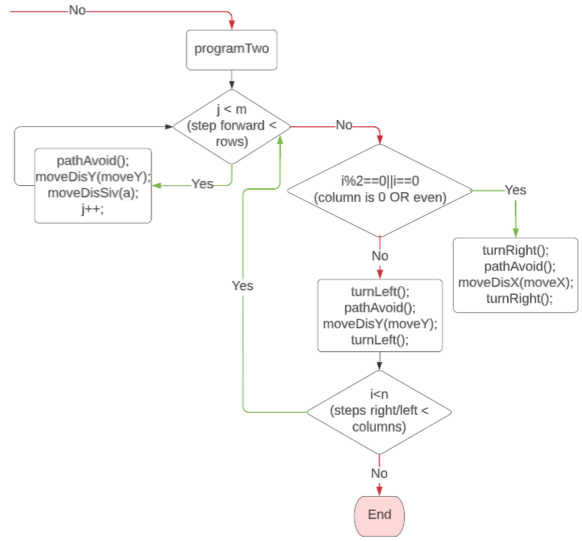

Fig. 15: Overview of programTwo flowchart

1) Terrain input

As soon as ReGenBot's program is initialized, the robot asks the user to input the width and the length of their terrain in meters. Once this information is saved in the robot's memory, it asks the user to input the number of rows and columns they wish to map out the terrain and mark collection points at the intersections of the rows and columns. This user input is utilized in level one and level two of the program as it sets where the robot stops to collect soil samples and distribute fertilizer, respectively. The serial function parseInt (Fig. 16) is implemented to read the integer the user inputs at the beginning and store it in the respective variables. Since this is executed independently of the level selected for the robot, it is coded outside of the program switch cases.

```
Serial.println("Width of the terrain in meters: "); //prompt for input
while (Serial.available() == 0) {} //wait for input
x = Serial.parseInt(); //read user input and hold in x
Serial.println("Length of the terrain in meters: "); //prompt for input
while (Serial.available() == 0) {} //wait for input
y = Serial.parseInt();
Serial.println("Desired columns: "); //prompt for input
while (Serial.available() == 0) {} //wait for input
n = Serial.parseInt();
Serial.println("Desired rows: "); //prompt for input
while (Serial.available() == 0) {} //wait for input
m = Serial.parseInt();
```

Fig. 16 Terrain input code display

2) Program level input

ReGenBot is composed of two levels. This is initially recorded according to the input on the slide switch. If the switch is at a low state, to the left, it executes program one, the testing phase. Else if the switch is to the right, it executes program two, the distribution phase. (Fig 17)

```
int readSwitch = 0;
readSwitch = digitalRead(switchL);
if(readSwitch == LOW)
{
  programOne();
}
else
{
  programTwo();
}
```

Fig. 17 Read switch input

The first level is targeted to collect soil samples throughout the intersections of the rows and columns desired by the user. For this, void function programOne is called and executed. (Fig. 18) In this function, the robot moves in the positive Y direction until the number of rows is met. After this, the robot turns right or left depending on the column number. If the column is zero or even, it will turn right while if the column is odd, it will turn left. After the robot reaches each row point throughout the column, it executes the probeSensors function that lowers all the sensors into the soil, records it into their respective text files, and saves it to the SD card.

```
void programOne() //testing
{
int i = 0;
int j = 0;
/*----MOVING THROUGH THE TERRAIN-----*/
while (j<m)
{
  pathAvoid();
  moveDisY(moveY);
  j++;
  moveDisProbe(b);
  probeSensors();
}
if((i%2==0||i==0))
{
  turnRight();
  pathAvoid();
  moveDisX(moveX);
  turnRight();
  i++; j=0;
}
else if((i%2!=0))
{
  turnLeft();
  pathAvoid();
  moveDisX(moveX);
  turnLeft();
  i++; j=0;
}
if (i<n)
{
  Stop();
}
delay(500);
/*-------MOVING THROUGH TERRAIN END -----*/
}
```

Fig. 18 programOne function

Level two (Fig. 19) is targeted to distribute fertilizer or treatments that balance out the composition deficiencies in the soil. Similar to the code syntax followed for level one, the robot moves along the positive Y direction and trans left or right depending on the number of the column it is at. Once it reaches a row intersection, the robot stops and drops the fertilizer, or treatment, by executing the moveDisSiv function that has a nested while loop to control the sieve motor at the top base of the robot. This causes the robot to move the sieve from a locked position to an unlocked one, traveling about 1cm. Once it is unlocked, it travels the same distance in reverse to lock.

```
void programTwo() //distribution
{
  int i = 0;
int j = 0;
/*----MOVING THROUGH THE TERRAIN-----*/
while (j<m)
{
  pathAvoid();
  moveDisY(moveY);
  j++;
  moveDisSiv(a);
}
if((i%2==0||i==0))
{
  pathAvoid();
  turnRight();
  moveDisX(moveX);
  turnRight();
  i++; j=0;
}
else if((i%2!=0))
{
  pathAvoid();
  turnLeft();
  moveDisX(moveX);
  turnLeft();
  i++; j=0;
}
if (i<n)
{
  Stop();
}
delay(500);
/*-------MOVING THROUGH TERRAIN END -----*/
}
```

Fig. 19 programTwo function

3) Sensor data collection

Sensors are activated for level one of ReGenBot's system. the information for the temperature (Fig. 20), NPK, moisture, and pH (Fig. 21) data collection it's grouped up into the probeSensors function. Within this function, each criterion is read according to the sensor and saved in an individual file in the SD card. This allows the user to display the data in a graph for soil analysis. Similar to the movement the sieve performs, the moveDisProbe powers an encoded motor that inserts the sensors about six cm into the soil. (Fig. 22)

```
void probeSensors()
{
  //temperature
  regenbot = SD.open("temp.txt", FILE_WRITE);
  now = rtc.now();
  regenbot.print(now.hour());
  regenbot.print(":");
  regenbot.print(now.minute());
  sensors.requestTemperatures();
  Celcius=sensors.getTempCByIndex(0);
  Fahrenheit=sensors.toFahrenheit(Celcius);
  regenbot.print(" C  ");
  regenbot.print(Celcius);
  regenbot.print(" F  ");
  regenbot.println(Fahrenheit);
  delay(1000);
  regenbot.close();
```

Fig. 20 probeSensors function

```
//moisture
regenbot = SD.open("moist.txt", FILE_WRITE);
now = rtc.now();
regenbot.print(now.hour());
regenbot.print(":");
regenbot.print(now.minute());
output_value= analogRead(sensor_pin);
output_value = map(output_value,550,0,0,100);
regenbot.print("Mositure : ");
regenbot.print(output_value);
regenbot.println("%");
delay(1000);
regenbot.close();
```

```
 //npk sensor
 byte nitrogen_val,phosphorus_val,potassium_val;
 regenbot = SD.open("npk.txt", FILE_WRITE);
 now = rtc.now();
 regenbot.print(now.hour());
 regenbot.print(":");
 regenbot.print(now.minute());
 nitrogen_val = nitrogen();
 delay(250);
 phosphorus_val = phosphorous();
 delay(250);
 potassium_val = potassium();
 delay(250);
 //print
 regenbot.print("Nitrogen_Val: ");
 regenbot.print(nitrogen_val);
 regenbot.println(" mg/kg");
 regenbot.print("Phosphorous_Val: ");
 regenbot.print(phosphorus_val);
 regenbot.println(" mg/kg");
 regenbot.print("Potassium_Val: ");
 regenbot.print(potassium_val);
 regenbot.println(" mg/kg");
 delay(2000);
 regenbot.close();
}
```

Fig. 21 NPK and moisture sensor configuration for data collection

```
void moveDisProbe (int b) //testing motor
{
  //ticks per m = 763.94
  int ticks ;
  ticks = b*763.94;
  while (encA2[1]<ticks)
  {
    moveProbe();
  }
}
```

Fig. 22 moveDisProbe function

4) *Obstacle avoidance*

The ultrasonic sensor sends out a pulse and if an object is detected within 19cm (Fig. 23) of the robot, the motors decrease their speed and change their path. Once the robot stops, the servo motor rotates the ultrasonic sensor from right to left to send signals once again and detect which way is best to go. This is done through the compare Distance function, which chooses to turn right or left depending on which side detects the furthest distance from an obstacle.

```
serv.write(80); //servo position at 80 deg
delay(100);
distance = search();

if (distance < 19) //object is close
{
analogWrite(PWM[0],100); //decrease the speed
analogWrite(PWM[1],100);
changePath();
}

else if ((distance >= 19) && (distance < 60)){ //object is far
analogWrite(PWM[0],200); //increase the speed
analogWrite(PWM[1],200);
forward();  //move forward
}
```

Fig. 23 Path avoidance code

5) *Path correction*

Once the ReGenBot changes its path, it travels at an angle until another obstacle is detected in its way. For the robot to continue a straight path, a path correction function is implemented after the robot turns. (Fig. 24) For example, the function backToPathR is implemented to direct the robot that changed path and turned right, back to the original path (straight). For this, the ultrasonic sensor turns to the opposite direction its wheel turns and runs a similar syntax code to the path avoidance function, to check if the object is still in sight. The speed implemented for the path correction is low to decrease the probability of the robot overshooting. The number of steps that the robot takes is incremented while the distance to the obstacle is less than 19cm. Once the obstacle is far away, the robot turns in the opposite direction and takes the same number of steps. Once the number of steps is equivalent, it turns in the opposite direction one more time to continue along the column.

```
void backToPathR(){
  int numberSteps = 0;
  int i =0;
  while (distance < 19) //object is close
  {
  analogWrite(PWM[0],100); //decrease the speed
  analogWrite(PWM[1],100);
  numberSteps++;
  }
  turnLeft();
  for (i=0; i<numberSteps; i++)
  {
  analogWrite(PWM[0],100); //decrease the speed
  analogWrite(PWM[1],100);
  }
  turnRight();
}

void backToPathL(){
  int numberSteps = 0;
  int i=0;
  while (distance < 19) //object is close
  {
  analogWrite(PWM[0],100); //decrease the speed
  analogWrite(PWM[1],100);
  numberSteps++;
  }
  turnRight();
  for (i=0; i<numberSteps; i++)
  {
  analogWrite(PWM[0],100); //decrease the speed
  analogWrite(PWM[1],100);
  }
  turnLeft();
}
```

Fig. 24 Path correction code

VII. Product Evaluation

The current state of development for the ReGenBot is to receive funding and begin the production of a physical prototype. After the final assembly of the prototype, tests will be conducted in New York forests of different terrains to assess the readings and mobility of the robot to make necessary changes or adjustments. Then the ReGenBot will be sent with the team to Bolivia and Colombia to test the performance of the soil nutrient data collection with possible collaboration with Armonia, a Bolivian conservation non-governmental organization.

VIII. Conclusion

ReGenBot is a robot design that focuses on soil revitalization and data collection of burned forests. It decreases labor cost and time by more than half, and its low cost compared to the market makes it affordable for organizations and communities in South America. The user-robot interaction is simple and straightforward, making the robot accessible to many individuals in the audience. ReGenBot reduces human interaction with the delicate and dangerous state of forests while incrementing its survival rate.

For future work, new features can be implemented to prepare ReGenBot to be a multipurpose agricultural robot. Artificial Intelligence could be developed to make decisions based on the collected data, where the device could scan the terrain and display a map of the surroundings, identifying the amount of burned wood, the color of the soil, and signs of wildlife. With time, ReGenBot can be repurposed to conduct the replantation and regrowth process by spreading seeds and water while traveling through the area.

Acknowledgment

The authors would like to thank the President of Vaughn College of Aeronautics and Technology, Dr. Sharon DeVivo, along with the Engineering and Technology Department Chair, Dr. Hossein Rahemi, the Department of Education federal grant (Title III, Part F, HSI-STEM, and Articulation grant) which provided necessary funding support to engage Vaughn's students in STEM-related scholarly and professional activities. and faculty professor Dr. Miguel Bustamante for their continued support throughout the whole project. Also, a special thank you to Professor Donald Jimmo for his continuous support in putting this project together.

References

[1] Butler, R. A. "Rainforest Information." *Mongabay.* August 2020. [Online]. Available: https://rainforests.mongabay.com/deforestation/archive/Total_South_Am erica.htm. [Accessed: January 21, 2022].

[2] Arenteras, D., Dávalos, L. M., Barreto J. S., Miranda A., Retana J., et al. "Fire-induced loss of the world's most biodiverse forests in Latin America." *ScienceAdvances.* DOI: 10.1126/sciadv.abd3357. August 13, 2021. [Online.] Available: https://www.science.org/doi/10.1126/sciadv.abd3357. [Accessed: February 7, 2022].

[3] Harvey, F. "Amazon near tipping point of switching from rainforest to savannah – study." *The Guardian.* October 5, 2020. [Online]. Available: https://www.theguardian.com/environment/2020/oct/05/amazon-near-tipping-point-of-switching-from-rainforest-to-savannah-study. [Accessed: February 9, 2022].

[4] Minas. S., "How Do Wildfires Affect Soil?" *AES, C.M. APPLIED EARTH SCIENCES.* November 12, 2019. [Online]. Available: https://www.aessoil.com/how-do-wildfires-affect-so. [Accessed: February 9, 2022].

[5] Naresh. R., "What Are the Ideal Levels of NPK in Soil in Ppm?" *ResearchGate.* March 10, 2022. [Online]. Available: https://www.researchgate.net/post/What-are-the-ideal-levels-of-NPK-in-soil-in-ppm. [Accessed: May 5, 2022].

[6] "TB67H420FTG Dual/Single Motor Driver Carrier." *Pololu.* [Online]. Available: https://www.pololu.com/product/2999