Rufus: The Grand Concept

Created by: Coy Beardmore

Assisted by: Griffin McConaha, Sawyer Ramsey, Elija Cox, Joshua Lynch

Documented by: Sawyer Ramsey and Griffin McConaha

The Reason Why Rufus Was Started

The CIM class was challenged to make an Automated Guided Vehicle that can scale stairs and round the layout of the school’s interior.

The idea was to mimic the design of Spot, who is Boston Dynamics’ robotic dog. This would allow for Rufus to take advantage of the dynamic design and make it climb stairs.

There were a few limitations discovered during the R&D process:

- Motors overheat
- Small battery life
- Overweight frail components
- Low powered motors
- The backlash of the motors 0.
- Weak microcontroller, constantly crashing
- Slow response time
- Not open source components, only vex parts

Coding Process

The first place to start with the code is to make the kinematic model. This simplifies the programming, instead of controlling 12 axes at the same time there are only 4 coordinates to program.

Kinematic model:

```
void FLLegPos(double h, double l, double B) {
 T = 7;      // thigh length
 S = 5.375; // shin length
 x = 0;      // distance between foot and hip
 F = 7.29;  // offset triangle
 b = 0;      // inner hip angle (h)
 y = 0;      // imaginary triangle hypotenuse (FruntRightue)
 lf = 0;     // l + F
 a = 0;      // a plus h = H

 // hip angle
 LegLengthHips = sqrt((B * B) + (h * h));

 FruntLeftHipDegree =
     (acos(((LegLengthHips * LegLengthHips) + (h * h) - (B * B)) /
           (2 * LegLengthHips * h))) *
     RdToDegree;

 if (B < 0) {
   FruntLeftHipDegree *= -1;
 }

 // leg angle
```

```
 x = sqrt((h * h) + (l * l));

 b = acos(((x * x) + (T * T) - (S * S)) / (2 * x * T));

 lf = l + F;

 y = sqrt(((h * h) + (lf * lf)));

 a = acos(((x * x) + (F * F) - (y * y)) / (2 * x * F));

 FruntLeftThighDegree =
     ((acos(((x * x) + (T * T) - (S * S)) / (2 * T * x))) + a) *
RdToDegree *
     ThighGearRatio;

 FruntLeftKneeDegree = ((acos(((S * S) + (T * T) - (x * x)) / (2 * T *
S))) *
                       RdToDegree * -1 * KneeGearRatio);

 FruntLeftThighDegree =
     FruntLeftThighDegree - (ThighHomeOffset * ThighGearRatio);

 FruntLeftKneeDegree = FruntLeftKneeDegree + (KneeHomeOffset *
KneeGearRatio);

 FruntLeftHipDegree = FruntLeftHipDegree * ThighGearRatio;
}
```

The x, y, and z coordinates are inputted into the void and the angles of all three axes are outputted while taking into account all the offsets of the joints zeros

Now that the tedious work is being done by the computer, a pre-programmed path can be followed for all the feet to make the robot walk. That path looks like this:

```
void RCWalk() {
 // Walk IN PLACE
 while (Controller1.Axis1.position() == 0 && Controller1.Axis2.position()
      == 0) {
   if (Controller1.ButtonA.pressing () == true) {
     Gyro.calibrate();
     wait(2.5, seconds);
   }
   //PLANT FEET
   //FL & BR
   CalcBallanceXAndYs();
   G1(10, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
      0);
   //move alternate feet for ballance
   //FR & BL
   CalcBallanceXAndYs();
   G1(11, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
      0);

   //Check for Estop
   waitUntil(RufusSleeping == false);

   //MOVE FEET UP
   //FL & BR
   G1(10, 0, 0, 12 - StepHeight, 100, 3, 100, 0, 0);
   //move alternate feet for ballance
   //FR & BL
   CalcBallanceXAndYs();
   G1(11, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
      0);

   //present wait timing
   wait(0.35, seconds);
```

```
//Reed center of gravity offset
//FindCenterOfGravity(10);

//Check for Estop
waitUntil(RufusSleeping == false);

//PLANT FEET
//FL & BR
CalcBallanceXAndYs();
G1(10, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
   0);
//move alternate feet for ballance
//FR & BL
CalcBallanceXAndYs();
G1(11, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
   0);

//Check for Estop
waitUntil(RufusSleeping == false);

//present wait timing
wait(0.2, seconds);

//PLANT FEET
//FR & BL
CalcBallanceXAndYs();
G1(11, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
   0);
//move alternate feet for ballance
//FL & BR
CalcBallanceXAndYs();
G1(10, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
   0);

//Check for Estop
waitUntil(RufusSleeping == false);

//MOVE FEET UP
//FR & BL
```

```
  CalcBallanceXAndYs();
  G1(11, 0, 0, 12 - StepHeight, 100, 3, 100, 0, 0);
  //move alternate feet for ballance
  //FL & BR
  CalcBallanceXAndYs();
  G1(10, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
     0);

  //present wait timing
  wait(0.35, seconds);

  //Reed center of gravity offset
  //FindCenterOfGravity(11);

  //Check for Estop
  waitUntil(RufusSleeping == false);

  //PLANT FEET
  //FR & BL
  CalcBallanceXAndYs();
  G1(11, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
     0);
  //move alternate feet for ballance
  //FL & BR
  CalcBallanceXAndYs();
  G1(10, 0 + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100, 3, 100, 1,
     0);

  //Check for Estop
  waitUntil(RufusSleeping == false);

  //present wait timing
  wait(0.2, seconds);
}

// Walk FORWARD
while (Controller1.Axis2.position() > 0) {
  //Check for Estop
  waitUntil(RufusSleeping == false);
```

```
//PLANT FEET
//FL & BR
CalcBallanceXAndYs();
G1(10, StridDistance + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100,
   3, 100, 1, 0);
//move alternate feet for ballance
//FR & BL
CalcBallanceXAndYs();
G1(11, (StridDistance * -1) + AddBallanceFootX, 0 + AddBallanceFootY,
   12, 100, 3, 100, 1, 0);

//Check for Estop
waitUntil(RufusSleeping == false);

//present wait timing
wait(0.35, seconds);

//Check for Estop
waitUntil(RufusSleeping == false);

//MOVE FEET UP
//FL & BR
CalcBallanceXAndYs();
G1(10, StridDistance * -1, 0, 10, 100, 3, 100, 0, 0);

//MOVE FORWARD
//FR & BL
CalcBallanceXAndYs();
G1(11, StridDistance + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100,
   3, 100, 1, 0);

//Check for Estop
waitUntil(RufusSleeping == false);

//present wait timing
wait(0.35, seconds);

//Check for Estop
waitUntil(RufusSleeping == false);
```

```
//PLANT FEET
//FL & BR
CalcBallanceXAndYs();
G1(10, (StridDistance * -1) + AddBallanceFootX, 0 + AddBallanceFootY,
   12, 100, 3, 100, 1, 0);
//move alternate feet for ballance
//FR & BL
CalcBallanceXAndYs();
G1(11, StridDistance + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100,
   3, 100, 1, 0);

//Check for Estop
waitUntil(RufusSleeping == false);

//present wait timing
wait(0.35, seconds);

//Check for Estop
waitUntil(RufusSleeping == false);

//MOVE FEET UP
//FR & BL
CalcBallanceXAndYs();
G1(11, (StridDistance * -1), 0, 10, 100, 3, 100, 0, 0);

//MOVE FORWARD
//FL & BR
CalcBallanceXAndYs();
G1(10, StridDistance + AddBallanceFootX, 0 + AddBallanceFootY, 12, 20,
   3, 100, 1, 0);

//Check for Estop
waitUntil(RufusSleeping == false);

//present wait timing
wait(0.35, seconds);

//Check for Estop
waitUntil(RufusSleeping == false);
```

```
  //PLANT FEET
  //FR & BL
  CalcBallanceXAndYs();
  G1(11, (StridDistance * -1) + AddBallanceFootX, 0 + AddBallanceFootY,
     12, 100, 3, 100, 1, 0);
  //move alternate feet for ballance
  //FL & BR
  CalcBallanceXAndYs();
  G1(10, StridDistance + AddBallanceFootX, 0 + AddBallanceFootY, 12, 100,
     3, 100, 1, 0);

  //Check for Estop
  waitUntil(RufusSleeping == false);
}
```

The gyroscope is constantly read for balance through this void:

```
void CalcBallanceXAndYs () {
/*REFRENCE:    (as Rufus is going "down" in that direction)
                            /\
                            |   (-Roll direction)
                            |
                       FL-------FR
                       |         |
(+Pitch direction)  <--  |         |   -->  (-Pitch direction)
                       |         |
                       |         |
                       BL-------BR
                            |
                            |   (+Roll direction)
                            \/
*/

 AddBallanceFootX = Gyro.orientation(roll, degrees) * GyroSensitivity;
 AddBallanceFootY = Gyro.orientation(pitch, degrees) * GyroSensitivity;

 if (AddBallanceFootX > 2) {
   AddBallanceFootX = 2;
 }
 if (AddBallanceFootX < -2) {
   AddBallanceFootX = -2;
 }
 if (AddBallanceFootY > 2) {
   AddBallanceFootY = 2;
 }
 if (AddBallanceFootY < -2) {
   AddBallanceFootY = -2;
 }
```

```
 //sum offsets
 //SumBallanceVars();

 //BLAddBallanceFootZ = (sin(Gyro.orientation(pitch, degrees) +
Gyro.orientation(roll, degrees))) * (7.1875);
}
```

Hardware

Rufus is built on the VEX hardware platform. The V5 kits specifically were used in Rufus’s construction and programming. There were also some older IQR sensors used.

It is ideal to make the robot as light as possible, this makes it more powerful and efficient. So, as many pieces as possible should be made of aluminum. Because it's lighter and relatively rigid in comparison to the other options.

Though one of the main problems with the final prototype is that the aluminum continues to bend and the more it's bent the easier it is to be bent again. However, for the sake of prototyping aluminum was used to make it as light as possible.

During the design of Rufus, there were several choices made to improve him. Rufus started with some smaller feet than the final design. Wider feet were added to offer greater stability while standing and moving. Rufus was originally designed only with axles and motors to move all of his parts. This was changed in the final design to turntables because the axles would bend and twist while Rufus was attempting to walk. For the Knees, there was a need to use 2 motors per knee joint because more power was needed to keep the robot upright. In the construction of Rufus,

lock nuts were used. In the initial trials of making Rufus walk, he would commonly have his bolts unscrewing. Another thing needed to improve upon was the turntables. They were not commonly used in the class so they made a terrible noise and some of them were tighter than others. This was fixed by applying some lubrication to the turntables, WD-40. The inner gears also had to be changed out of the turntables because the original plastic gears were not able to hold up to the power of the dual motors that were running them. Metal gears were ideal for the forces being applied to the joint.

Motor Problems:

The VEX kits that were used in this project were brand new at the time that the project started. The VEX V5 kits come with 4 motors and they can be exchanged to be able to run high speed, mixed, and high torque gearboxes. This allows for some flexibility. The motors are constructed out of brittle plastic that insulates the heat instead of dispersing it. What appears to be a vent on the side serves no purpose. They often overheat and lose much of their power in the overheated state. This meant that Rufus was only able to run for about 5 minutes before he would have to cool off for around 20 minutes. This limited many tests. 3D printed housings were prototyped in an attempt to modify the housing. The plan was to either add a fan or an aluminum heat sink. At one point there was an attempt to apply Peltier modules directly to the housings to cool down the housings in hopes of extending the life of the motors. None of these attempts were successful. The Peltier modules were ineffective due to the insulative properties of the vex motor housing. The project came to a stop when the 3d printed motor housing idea was being executed. The final design does not have a fix for the motors. Rufus is only able to run for about 5 minutes before he has to rest and allow the motors to cool off.

A possible fix to this may be to add a heat sink that is exposed to the open air to allow the motor to disperse heat.

Reason For Using Multiple Brains:

During the original phases of the project, Rufus was run off of one brain that would handle everything. Rufus has now been upgraded with two more brains to a total of three brains. One of them is the manager , up front, while the other two are the workers. The front brain handles most of the walking patterns, talking to the controller, and is partially responsible for keeping the left and right side in sink. It communicates all of this through G-code. The workers do the hard kinematic calculations to tell the motors where to go. After this, the workers are also paired together outside of the manager brain, this is called the union link, the main purpose of the union link is so that the left and right side stay in sink since the separate brains control the left and right side. Through all of this, the battery length was extended greatly. Each brain requires a battery to run so the power is dispersed among the robot. This results in three times the battery life. It also in turn spreads out the workload of the microcontrollers. With just one microcontroller there were points where the brain would get behind and was not able to keep running Rufus. With the three controllers being running all at the same time it decreased the latency of the calculations to the legs.

Roll cage:

The roll cage was designed so that if the Rufus rolled over no brains or batteries would be damaged. The roll cage also has E-stop buttons mounted on the very top so when he rolls over all of the programs shut down so no further damage is incurred.

Ports:

When starting construction there was only one brain which was limited in terms of I/O. There were only 21 ports for the new V5 sensors/motors. There are also only eight ports for 3-wire accessories to be added on. With the single brain, many issues arose purely because of the lack of ports. With three brains Rufus was able to have everything that he needed and some ports left over for expandability. 3-wire expanders were also used.

Color Codes:

During Rufus' design phase it was easy to get confused with the wiring even if they were managed because of the sheer amount of wires within the chassis. The fix to this was a color system for the wires. There is yellow tape on the left side of Rufus and the right side of Rufus is red.

Safety features. - estop, on the death of the brain, LEDs Rufus has several safety precautions built in to protect itself and the people around it. As discussed earlier, there are E-stop buttons built into the roll cage. This protects against accidental rollovers so the robot does not cause further damage to itself or anyone around it. The robot also has protection to where if one of the brains battery dies the entire system will shut down.

Load Cells:

The Load Cells was an idea to maintain equal weight across all four of the feet. However, the implementation of this was not as easy as predicted. There were a few problems with that. Number one, the Vex brain wouldn’t talk to the load cells. The idea was abandoned for a Vex made solution.

Limit switches were installed to all the feet in order for the brains to know when the feet were on the ground. If the brain tried to pick a foot off the ground and it didn’t then the brain knows there's too much weight on that leg. The concept was to shift the weight away from that leg to obtain balance. Though the program didn’t end up working in a reasonable amount of time and the project was brought to a conclusion and presented as a concept. Though these ideas are planned to be implemented in the future.

Future plans

With new technologies, Rufus could be improved. There are few new Vex V5 technologies coming soon that would allow Rufus to do cool new things such as GPS guided walking. The original goal of navigating the school was going to be solved by an AI that was thought of. It would use it to map out the layout of the building in a wandering mode that would tell elevation and record it based on where Rufus was. The AI would figure out the most efficient path from one place to another using pathfinding methods. A little bit of research was done into this but it was quickly found out that V5 was not going to be able to do this. A special thank you to Tuscarawas Valley Local Schools Project Lead the Way program for the resources, the online Vex Forum for assistance on Vex Link, Sawyer Ramsey for the documentation, Paul Dunlap for guiding the project, and everyone who assisted along the way!

Rufus' Talents:

With the initial design of Rufus, he needed to be able to scale an incline like stairs and a ramp. He also needed to be able to walk for a large distance. In the end, he was able to walk but climbing the stairs still eludes him. Besides the basics he is also able to sit like a dog, lie down, he can bow, he can walk in the place, walk , crawl on the ground, dynamically balance, and do a little dance. Rufus has rollover protection with E-Stop buttons on his top. All of these things are controlled by a remote controller that an operator is controlling.

Rufus Specs:

Weight: 21Lbs

Height: about 2ft standing

Length: 2ft

Width: 1.5ft

Brains: 3 VEX V5

Motors: 16 VEX motors (4 of them are high torque for the hip axis motors while the rest are normally geared)

Batteries: 3 1100 mAh (Totalling 3300 Mah)