ROMO – Robotic Autonomous Lawn Mower

Kevin Moriarty, CSE, Aaron Stam, EE, Collin Timmerman, EE, and Leonardo Luchetti, EE

***Abstract*— We introduce ROMO (Robotic Mower), an autonomous lawn mower that will allow users to relax while a robot mows their lawn for them. The system uses real time kinematic GPS to precisely position itself within the user's lawn. It turns by varying the speed of its two rear drive wheels allowing it to mow into corners.**

I. Introduction

The average American spends about 70 hours a year on lawn care, according to the American Time Use Survey. [1] This time, considered to be a chore, could be spent more productively on other tasks. Additionally, there is a level of physical effort required to operate a standard push lawn mower and can result in injuries especially to the lower back. If a person does not possess the physical ability to mow, they are left with two options, purchase an expensive riding mower, or pay for a lawn care service. There currently exist several autonomous lawn mowers on the market. The problem with this solution is that they cost anywhere from $1000 to $3000[2], require time consuming setup, and have poor battery life, requiring multiple charges to move a lawn. Similarly, a lawn care service can cost approximately $1000 per year.

Our system requirements have been developed based on the requirements of an average consumer. As such we have determined that the mower should be able to mow a 1500 sq. ft. lawn on a single battery charge. The optimum speed for mowing a lawn is 2.5 to 3.5 miles per hour. [3] As such we have targeted a mowing speed of 3.5 mph. In order to mow the lawn with a hypothetical 12-inch blade we have determined that we need to be able to resolve the relative position of the mower with an accuracy of 5cm or better. In order to simplify the initial design, we are assuming a rectangular, level, obstacle free lawn with a known starting position.

Requirement	Specification
Lawn Area	1500 sq. ft.
Mowing Speed	3.5 +/- 1.0 mph
Battery Life	1 charge = 1500 sq. ft.
Position Accuracy	Better than 5cm

Table 1: List of System Requirements and Specifications.

II. Design

A. System Overview

Our initial approach to system design was to focus on our positioning system as we believed that it would be the most challenging aspect of the project. As mentioned we need a high level of both accuracy and precision in our positioning system. However, because of the locality of a lawn it is not necessary to know this position relative to global coordinate systems. Instead it may suffice to know the precise position of the rover within the locality of the lawn. Given this it is still necessary to have a reference position from which the rovers position will be known. In our system that reference point is a stationary base station.

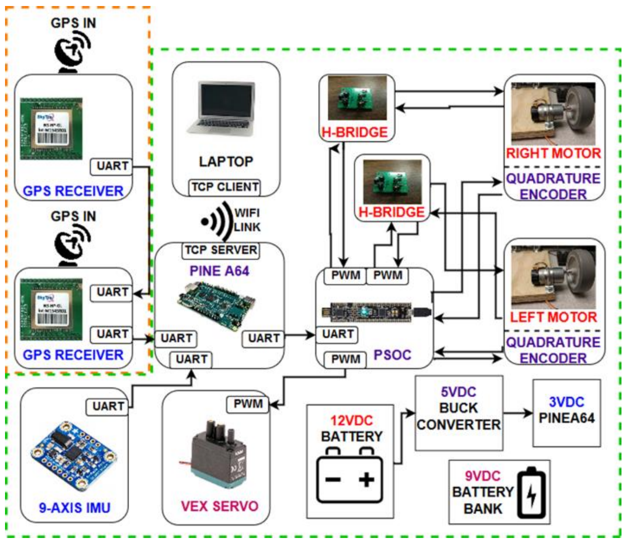


Figure 1: RoMo Block Diagram

The system used to measure the relative position of the rover relative to the base station is an advanced form of GPS known as Real Time Kinematic (RTK) GPS. This form of GPS uses measurements of the phase of the GPS carrier signal at multiple receivers to determine the relative position of the receivers. [4,5] In our system the base station generates measurements with are forwarded to the rover where they are used to calculate the displacement between the base station and the rover.

The GPS system also has limitations. Firstly, it only updates position once per second. [4,5] Given the prescribed mowing speed the rover would travel 1.5 meters between updates. This may be acceptable for long strait paths however it is not acceptable when making turns. Furthermore, a connection between the base station and rover must be maintained continuously for the GPS system to function correctly. [4,5] If the connection was interrupted the rover would have no way to correct its position.

Given the above limitations of the GPS system it is necessary to augment its output between updates. This is accomplished using a dead reckoning system. The dead reckoning system uses measurements from a gyroscope as well as measurements of wheel speed. The gyroscope generates a measurement of yaw in degrees per second. [6] Integrating this measurement allows for the determination of the current heading. The wheel speed measurement determines the linear speed. Together this allows us to integrate the velocity vector to a displacement between GPS fixes. The advantage of this system is that the update rate of the wheel speed and yaw rate measurements are much higher allowing for the position to be determined between GPS updates.

As mentioned there is a need for a connection between the base station and the rover for the GPS system to function properly. This connection must be maintained continuously and must simulate a serial port operating at 57,600 baud. [4,5] Given these requirements it was proposed to used Wi-Fi to complete this connection. It was believed that Wi-Fi___33 would offer a stable proven connection between the base station and rover.

In order to traverse the lawn there must be some physical rover capable of locomotion within a 2-dimensional plane. In order to accomplish this a wood platform with three wheels was chosen. At the rear of the platform two drive motors were mounted which have built in encoders to determine wheel speed. At the front of the platform a steering wheel was mounted. This wheel can be rotated as it is mounted on an axle connected to a traditional RC servo motor. The drive motors are driven with Pulse Width Modulation (PWM) at 12V. The RC servo is supplied 9V and a control signal at 5V.

The above systems are useless without a computing system to run them. In order to simplify the software aspect of the project a two-component solution was chosen. A Pine A64 acts as the primary computer. It has a quad core processor and runs Linux. [7] A PSOC 5LP microcontroller is used as a real time processor to decode input signals and generate control signals. This setup allows the use of the best of what both technologies offer. The combined environment has the real time hardware capabilities associated with microcontrollers; along with the processing power and multithreaded capability of full size computers.

In order to power all of these systems a somewhat complex power distribution system is necessary. The rovers power system consists of two batteries and a buck converter. Additionally, there is another buck converter on the Pine A64. Overall there are 4 different supplies accessible on the rover. The base station has one battery and one linear voltage regulator.

B. RTK GPS System

Our requirement for the GPS system is to be able to determine the mowers differential position from the base station within 5 cm. Traditional GPS receivers are simply not accurate enough for this purpose. The extension of traditional GPS positioning that enables this level of position is known as real time kinematic GPS (RTK). Traditional GPS receivers calculate their position using the code information transmitted by multiple GPS satellites. [8] RTK GPS differs in that it uses the measurement of carrier phase. [8] This measurement leaves several unknown variables. These are the receiver clock offset, the satellite clock offset, the hardware biases, and the number of wavelengths between the satellite and the receiver.[7] If the carrier phase measurement is taken in two locations the first two unknown variables cancel leaving only the number of wavelengths between the satellite and receiver unknown.[8] By fixing these at an integer number it then becomes possible to calculate the differential position between the two receivers[8] If one of these receivers is in a fixed location the position of a mower in reference to this location becomes known.

The implementation of RTK GPS we are using consists of two SkyTraq S2525F8-GL-RTK RTK capable GPS receivers connected via a wireless link. The base station has a GPS module with outputs connection data via a serial port running at 57,600 baud. [4,5] This data is transmitted via a wireless link to the mower where it is input to the mower’s GPS receiver. The GPS receiver on the mower uses its own measurements as well as those from the base station to calculate its position relative to the base station. [4,5] The RTK receivers we are using are rated for “1cm+1ppm” accuracy. [13] This equates to accuracy relative to the base station of better than 2cm when within 10km of the base station. The mowers GPS module outputs the differential position over another serial port running at 115,200 baud to the Pine A64 up to once per second. [4,5] Results from a static test can be seen in figure 2. This shows the drift over 300 samples or 5 minutes is less than 2cm.

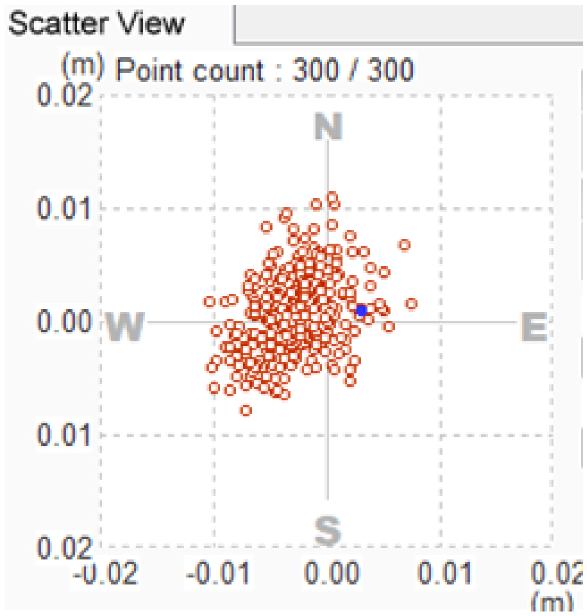


Figure 2: GPS Position over 5 Minutes

On board the Pine A64 incoming data is fed into a circular buffer. Then the ends of the messages are determined, and they are parsed into usable data. This parsed data is then timestamped and stored in a common data storage area to be used by other software components, namely the dead reckoning system.

C. Wireless Link

The wireless link was implemented on two NodeMCU 1.0 devkits, each containing an ESP8266 wireless module.[9,10] Websockets were chosen as the method of data transfer due to the high latency provided.[11,12] Websockets allow real-time transfer of data through TCP connections that are kept open, eliminating the overhead of establishing a connection every time data is to be transferred.[11,12] The NodeMCU attached to the mower is configured as a WebSocket server, and the NodeMCU comprising the base station along with the other GPS antenna is configured as a WebSocket client. The server establishes a wireless access point upon application of power and waits for a connection request from the client. The client when powered, continuously pings for the server's access point, and initiates a connection upon receiving a response.

Once the connection is established, any UART data that arrives to the serial port of the client NodeMCU is converted to binary data and sent over the wireless access point to the server NodeMCU, which reformats the binary data to UART and outputs it from the serial port on the server to the GPS module on the mower via cable. While we were successfully able to implement this system, it was not able to provide the latency or reliability required to function correctly. This was determined to be due to the strength of the wireless provided by the ESP8266 when operated without an intermediate router. Further, we determined that using an external router would not solve to issue. The reason for working with a self-contained system was so that the user would not be required to move their wireless router outside each time they wished to mow their lawn. Overall, the ESP8266 was found to be more suitable for applications not requiring 100% successful, real time packet transfer, such as temperature and humidity monitoring. When attempting high speed repeated transfers, the system drops packets. Further, the SDK only provides closed source binary libraries so using the modem without the Wi-Fi protocol is impossible. [12]

D. Gyroscope Control System

The gyroscope control system consists of both hardware and software components. The hardware aspect of the system is a Bosch BNO055 IMU which is connected to the Pine A64 computer via UART at 115,200 baud. [6] The software control runs on its own thread and retrieves the yaw measurement from the gyroscope in a signed integer format up to 100 times per second. [6] This information is then converted into a floating-point value in degrees per second. This value is then continuously integrated to give a relative heading since system start. (Equation 1) This value is then timestamped and exported to a common storage area for use by other parts of the system, namely the dead reckoning system.

$$\theta_t = \theta_{t-1} + \Delta\theta_t * \Delta t \qquad (1)$$

E. Wheel Speed Sensing

In order to determine the speed of the wheels of the mower we selected motors which have built in encoders. Each encoder generates 64 pulses per revolution. [13] Given the operating speed this means that there will be a maximum of 11,733 pulses per second. Such high speed makes in impractical to decode the signals using interrupts and software. This led to the use of the microcontroller with a hardware quadrature decoder, the PSOC. The PSOC is interfaced with the Pine A64 computer via UART at 115,200 baud. The software running on the Pine A64 requests the current decoder value over UART and the 32-bit value of each of the hardware decoder registers is returned. [14] The difference in position over time is then taken. This measurement of counts per second is then converted into a floating-point value of revolutions per minute. (Equations 2-3) The RPM value measured from each wheel is then timestamped and exported to a shared area for use by other parts of the system, namely the dead reckoning system. This is repeated at 30Hz.

$$CPS = \frac{\kappa_t - \kappa_{t-1}}{\Delta t} \tag{2}$$

$$RPM = \frac{CPS}{32} \tag{3}$$

F. Dead Reckoning System

The GPS system alone is not capable of providing the positioning information necessary to follow a path through the lawn. In order to augment this a dead reckoning system was constructed. It takes information from the GPS system, the gyroscope, and wheel speed sensors to allow for positioning updates at a much greater rate, up to 30Hz. Additionally, under certain conditions the GPS information may be inaccurate. An example of this is that GPS heading is calculated as the direction from the previous point to the current point. In a turn this may not be equal to the true direction of travel. In order to prevent this the dead reckoning system uses the last three GPS positions and only considers the GPS position valid when the second derivate of position is close to zero. That is the north and east velocity vectors are constant. This allows the dead reckoning system to reject inaccurate GPS heading information when negotiating turns.

In order to update position between accurate GPS fixes the gyroscope and wheel speed data are utilized. The wheel speed allows for the determination of linear speed given the known wheel diameter of 100mm. [15] When an accurate GPS heading is received the current heading from the gyroscope is saved. Every time there are updated values from both the gyroscope control system and the wheel speed sensing system the dead reckoning system determines the new heading and speed and integrates this velocity vector to determine the new position. (Equations 4-9)

$\theta_{GYR}(0) = Gyroscope\ heading\ at\ last\ valid\ GPS\ update$
$\theta_{GPS}(0) = last\ valid\ GPS\ heading$
$P_{North}(0) = North\ Displacement\ of\ last\ valid\ GPS\ fix$
$P_{East}(0) = East\ Displacement\ of\ last\ valid\ GPS\ fix$
$RPM_{avg} = wheel\ speed\ averaged\ between\ the\ two\ wheels$

$$V_{wheel} = \frac{RPM_{avg} * 0.1 * \pi}{60}\left(\frac{m}{s}\right) \tag{4}$$

$$\theta(t) = \theta_{GPS}(0) + \left(\theta_{GYR}(t) - \theta_{GYR}(0)\right) \tag{5}$$

$$V_{NORTH}(t) = V_{lin} * Cos\left(\theta(t)\right) \tag{6}$$

$$V_{EAST}(t) = V_{lin} * Sin\left(\theta(t)\right) \tag{7}$$

$$P_{North}(t) = P_{North}(t-1) + V_{North}(t) * \Delta t \tag{8}$$

$$P_{East}(t) = P_{East}(t-1) + V_{East}(t) * \Delta t \tag{9}$$

G. Motor Control

The two motors at the back of the vehicle as well as the steering servo are driven by the motor control system. This system takes the outputs of the path following system and generates PWM signals. The commanded motors values are first retrieved from a common shared data area. Then they are converted into units for the PSOC microcontroller. The commands are then sent to the PSOC over a 115,200 baud UART link. The PSOC generates two PWM signals which drive low side switches (Figure 3) for each of the rear motors. The modulation range is from 0 to 100%. It also generates a PWM signal to the front steering servo. The angle of the steering servo is determined by the length which the pulse is high. The range is 1ms to 2ms high time. Lengths less than 1.5 ms turn to the left where widths greater than 1.5ms turn to the right. [16]

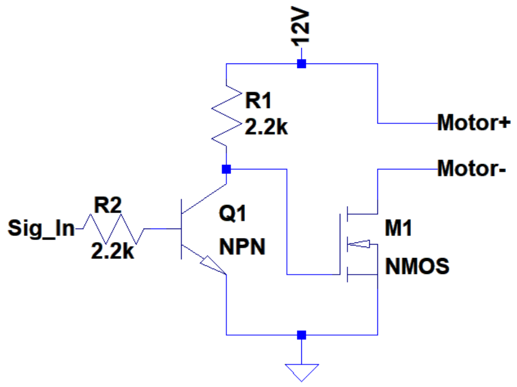


Figure 3: Low Side Schematic

H. Path Following

The function of the path following system is to adjust the steering angle in order to follow a predetermined path. In order to do this, it first imports the current position output of the dead reckoning system. It then determines the heading and range to the next position along the path. (Equations 10-13)

$P_{RN} = North\ Displacement\ of\ Rover$
$P_{RE} = East\ Displacement\ of\ Rover$
$P_{TN} = North\ displacement\ of\ target\ position$
$P_{TE} = East\ displacement\ of\ target\ position$
$\theta_{Cur} = current\ heading\ of\ rover$

$$R_{RT} = \sqrt{(P_{RN} - P_{TN})^2 + (P_{RE} - P_{TE})^2} \tag{10}$$

$$\theta_{RT} = \operatorname{atan}\left(\frac{P_{RE} - P_{TE}}{P_{RN} - P_{TN}}\right) \tag{11}$$

$$-180° < \theta_{RT} < 180° \tag{12}$$

$$\theta_{Steer} = C * (\theta_{RT} - \theta_{Cur}), C > 1 \tag{13}$$

The steering angle is determined from the heading error ($\theta RT - \theta Cur$) multiplied by a gain constant. Additionally, if the range to the point R_{RT} is less than a constant this point is considered to have been passed through. If this is the case the next iteration of path following will use the next point along the path. This continues until the end of the path is reached at which point the rover will stop.

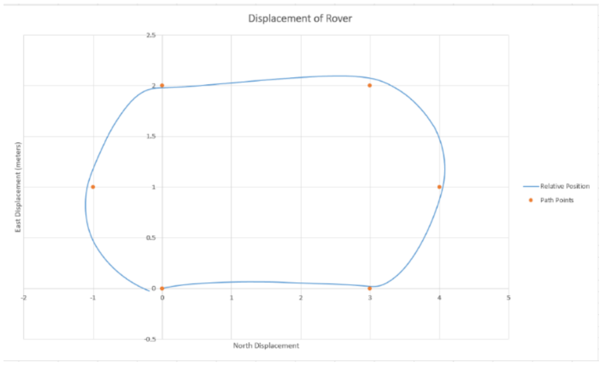


Figure 4: Rover following a generated path

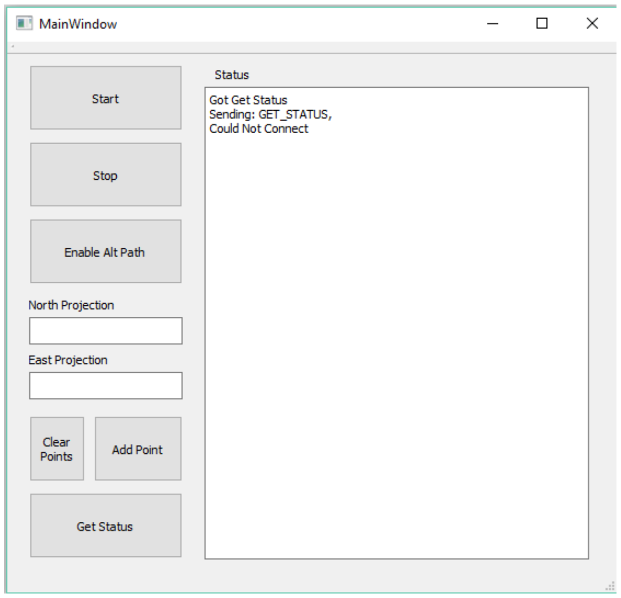
The above figure (Figure 4) shows the function of the path following system as measured with the GPS system disabled. The orange dots are the path points starting at (0,0) which the rover is to follow. Looking at the diagram we can see some drift between (0,0) and (3,0) this is the result of backlash in the steering system. The backlash means that commanding a small steering angle may not affect the direction of travel of the rover. The overshoot seen around point (3,2) is cause by the limited turning radius of this prototype.

I. User Interface and Control

Users must be able to interact with the rover to make it function. This function is accomplished via Wi-Fi. The rover runs a TCP command and control server on port 5555. The available commands are listed in table 2. Additionally, a basic client application was created for testing purposes which allowed easy execution of these commands via a graphical user interface. (Figure 5)

Command	Function
Start,	Begin following path from first point
Stop,	Disable path following, halt all motion
Add_Point, displacement_north, displacement_east,	Append point to end of stored path
Clear,	Clear stored path points
Enable_Path,	Enable the use of the stored path

Table 2: List of User Commands and Corresponding Functions



```
Got Get Status
Sending: GET_STATUS,
Could Not Connect
```

Figure 5: Rover Graphical User Interface

III. Project Management

FPR Deliverable	Progress
Travel 40 ft in straight line	Partial
Travel 40 ft in 25 seconds or less	Completed
Perform a 90-degree turn	Completed
Return to starting position	Partial

Table 3: A list of FPR Deliverables and progress

We managed to at least partially achieve all the goals we set for ourselves for FPR (Table 3), however there is still much work to be done to deliver a consistently functioning prototype. The development of RoMo was a bit of a rocky road. We managed to produce a prototype that was not tested to its full capabilities and embodies a tenuous definition of the word ‘functional’. This result was achieved largely due to conflicting schedules and lack of communication between members of our team. While we met with our advisor once a week, we did not meet nearly as often as a team although the frequency of our physical meetings increased as the project neared completion. There was friction between team members that resulted in the project not being fully tested and led to constant delays. We needed an external force to help hold team members accountable and aid in division of work. A team with varying skill levels and areas of talent, but little to no management or accountability via clear division of work wastes much effort in getting everyone on the same page or even getting team members to communicate or contribute to the project that could be spent advancing the project's state.

We also should have taken better advantage of external resources, such as technical experts to help us in challenging areas. We were unable to resolve the latency issues with our wireless link, not considering that there was a resident ESP8266 expert in M5. Aaron Stam was our team lead, lead programmer and tester, documentation and presentations second, and majority contributor. Kevin Moriarty was our programming and testing second, as well as being the lead for documentation, presentations, team communications, and the team website. Collin Timmerman constructed the RoMo chassis and mounted equipment, and Leonard Luchetti designed and ordered the H-Bridge PCBs which were modified for use in the demo day design.

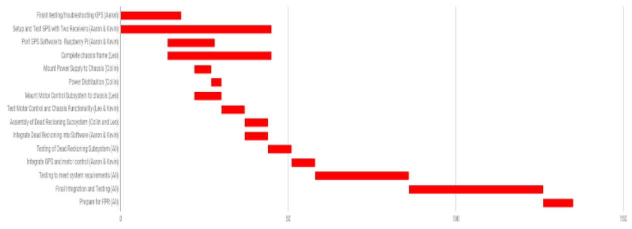


Figure 6: Gantt Chart Demonstrating Original Project Timeline

IV. Conclusion

In the time since MDR, almost all of the actual development of our project occurred. The mower chassis was constructed, parts were mounted, the microcontroller and CPU were successfully programmed and then tested after several changes in hardware and software. The microcontroller is successfully able to read input signals from the motor encoders as well as the gyroscope output from IMU unit, allowing for limited path following, with only this functionality the mower is unable to perform course corrections, it can only navigate a set of coordinates under the assumption of limited error, this is of course impractical for real world use in an autonomous rover. The installed GPS system is meant to provide the inputs for course corrections, and the functionality is coded into the final prototype, however testing was never performed extensively with the system functioning and we have no tangible results to verify the mower's ability to perform course corrections as of FPR. This lack of testing can be attributed to the lack of a working wireless link. The use of a wired connection between the base station and rover was attempted, however insufficient wire length lead to the rover dragging the base station behind it, eliminating any usefulness it may have added.

Future work would involve further testing and optimization as well as a redesign of the physical prototype to optimize weight distribution and install wheels better suited to grass, dirt, or similar surfaces. The problem of building a properly functioning wireless link for GPS correction data still remains. Appropriate hardware certainly exists that will resolve the latency and packet loss issues. Further exploration of the wireless link design could involve the use of a full-size computer hardware rather than low power IOT focused devices such as the original ESP8266.

VI. Appendix

A. Cost Analysis

Part	Quantity	Unit Price	Per 1000
Pine64 PC	1	$46.00	$38.40
GPS RTK Receiver	2	$80.00	$44.00
DC Motor/Encoder	2	$35.87	$35.87
12V DC Battery	1	$25.99	$18.99
IMU Chip	1	$27.88	$5.93
Back Wheels	2	$18.20	$7.88
Front Wheel	1	$5.99	$5.99
Vex Servo	1	$19.99	$19.99
PSOC	1	$10.00	$10.00
Buck Converter	1	$6.99	$3.50
H-Bridge	2	$6.83	$0.22
Total		$424.64	$278.74

VI. References

[1] C. Ingraham, "Lawns are a soul-crushing timesuck and most of us would be better off without them," in Chicago Tribune, chicagotribune.com, 2015. [Online]. Available: http://www.chicagotribune.com/news/opinion/commentary/ct-stop-mowing-your-lawn-20150805-story.html. [Accessed: Dec. 21, 2017]

[2] WORX WG794 Landroid Pre-Programmed Robotic Lawn Mower with Rain Sensor and Safety Shut-off : Garden & Outdoor

https://www.amazon.com/dp/B00SJEUFF4/. [Accessed: 21-Dec-2017]

[3] E. Perratore, “Cutting the grass in slow mow produces better results,” *Consumer Reports*, 10-Jun-2013. [Online]. Available: https://www.consumerreports.org/cro/news/2013/06/cutting-the-grass-in-slow-mow-produces-better-results/index.htm. [Accessed: 21-Dec-2017].

[4] “NS-HP-GL : GPS/GLONASS RTK Receiver,” *NavSpark*. [Online]. Available: https://navspark.mybigcommerce.com/ns-hp-gl-gps-glonass-rtk-receiver/. [Accessed: 05-Feb-2018].

[5] “S2525F8-GL-RTK : GPS/GLONASS RTK Reciever Module,” *NavSpark*. [Online]. Available: https://navspark.mybigcommerce.com/s2525f8-gl-rtk-gps-glonass-rtk-receiver-module/. [Accessed: 05-Feb-2018].

[6] “BNO055,” *Bosch Sensortec*. [Online]. Available: https://www.bosch-sensortec.com/bst/products/all_products/bno055. [Accessed: 05-Feb-2018].

[7] Pine A64 Single Board Computer Datasheet http://wiki.pine64.org/index.php/Main_Page#Datasheet. [Accessed: 18-Feb-2018]

[8] GMK, “RTK Fundamentals,” Sept 18, 2014. [Online]. Available: http://www.navipedia.net/index.php/RTK_Fundamentals. [Accessed Nov. 14, 2007].

[9] “NodeMCU Documentation.” [Online]. Available: https://nodemcu.readthedocs.io/en/master/en/. [Accessed: 05-Feb-2018].

[10] “ESP8266 Datasheet,” *Espressif Systems*. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. [Accessed: 05-Feb-2018].

[11] Links2004, “Links2004/arduinoWebSockets,” *GitHub*, 16-Oct-2017. [Online]. Available: https://github.com/Links2004/arduinoWebSockets. [Accessed: 05-Feb-2018].

[12] Espressif, “espressif/ESP8266_NONOS_SDK,” *GitHub*. [Online]. Available: https://github.com/espressif/ESP8266_NONOS_SDK/releases/tag/v2.1.0. [Accessed: 05-Feb-2018].

[13] CQRobot, “12V DC Geared Motor w/Encoder-366RPM+12kg.cm, Gear Ratio is 30:1,” AngelANQIOOO1GB datasheet.

[14]“PSoC® 5LP: CY8C58LP Family Datasheet.” *Cypress Semiconductor*, Cypress Semiconductor, 6 Mar. 2018, www.cypress.com/documentation/datasheets/psoc-5lp-cy8c58lp-family-datasheet-programmable-system-chip-psoc?source=search&cat=technical_documents.

[15] Devantech 100mm wheels w/ 5mm hub https://www.robotshop.com/en/100mm-diameter-wheel-5mm-hub.html. [Accessed: 27-Nov-2017]

[16] Vex EDR 3 Wire Servo Motor https://www.vexrobotics.com/motors.html. [Accessed: 25-Mar-2018]