Pancreas robot

by

Calvin Dahms

B.S., Kansas State University, 2018

A Thesis
submitted in partial fulfillment of the
requirements for the degree
Master of Science

Carl and Melinda Helwig Department of Biological and Agricultural Engineering

Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Approved by:

Major Professor

Dr. Daniel Flippo

Copyright

Abstract

The World continues to need increased production of food to sustain the expected population growth over the next 30 years. The global population is projected to be more than 9.7 billion people by 2050. These people will need to eat nearly double the amount of food that is produced today. There are many approaches to solving this food crisis dilemma and progress is being made on multiple fronts. Sources of production growth include larger and more precise machine technology; more robust fertilizer applications; and crop modeling for genetically enhanced phenotyping. This paper focuses on the use of robotic technology to help model crop growth by using an unmanned ground vehicle (UGV) for data collection.

Simple crop modeling helped the agricultural revolution in the 20th century. Today, scientists are building upon those models to create more complex ways to represent crops and their traits. These models require large amounts of data to observe and describe relationships between inputs and crop responses. This data needs to be dependable, consistent, and as close to the source as possible. To achieve that type of data for this project, a UGV was developed to traverse rugged field conditions. The UGV was designed to carry a Geophex Electromagnetic (EM) sensor that measures the electrical conductivity of the soil. This electrical conductivity will be used to decipher soil characteristics that underlie the growth potential of different wheat traits. The robot that carries the EM sensor must be designed to not interfere with the conductivity measurements of the sensor. The data collected must be accurate and repeatable.

The scope of this research project is to develop the UGV to carry the sensor through the harsh field environments while not interfering with the incoming EM signal from the sensor. The project also explores methods for the robot to navigate through its environment on its own to limit human influence on the recorded data.

Table of Contents

List of Figures vii

List of Tables xi

1 Global Food Needs for 2050 1

1.1 Globalization's Impact on Arable Land 2

1.2 Effects of Food Shortage and Insecurity 5

1.3 Production per Acre 8

1.4 Agricultural Responses to Growing Food Demand 8

1.4.1 Large Farm Equipment 9

1.4.2 Precision Agriculture 10

1.4.3 Fertilizing 11

1.4.4 Genotyping 13

1.5 Wheat Phenotyping 13

1.6 Wheat Sensor Integration 17

1.7 Robotic Autonomous Sensing 18

1.8 Geophex EM Sensor 19

1.9 The EPSCoR Project 24

2 Robot Design 26

2.1 The Field Environment 26

2.2 Initial Proposal 29

2.3 Environment Around the Sensor 30

2.4 Fabrication of Initial Prototype 31

2.5 Prototype Steering Redesign 34
2.6 Prototype Electrical Box 38
2.7 Prototype Testing 40
2.7.1 Servo Control 42
2.8 Structural Redesign 45
2.9 Updated Electrical Box 52
2.10 Wavelength Detection (Color) Sensor 57
2.11 Testing the Redesigns 60
2.12 Fabrication of the Second Robot 65
2.12.1 Serial Communication 68
2.13 Arduino Sensing and Communication 68
2.13.1 Compass Reading and Analysis 68
2.13.2 Power Sensing and Logging 69
2.14 LattePanda Advanced Programming 69
2.14.1 RTK Reading 70
2.14.2 Wavelength Detection and Image Processing 70
2.14.3 Object Detection Protocol 72
2.14.4 Object Distance Detection 76
2.14.5 Object Detection Path Planning 78
2.15 Energy Usage Data 80
2.16 Battery Capacity Data 85
3 Conclusion and Future Work 89
Bibliography 91
A Robot Component Specifications 95
B Code for the Pancreas Robot 100

B.1 Code for the MyRio 100

B.2 LattePanda 101

B.3 Code for the Arduino 109

B.4 Code for the Object Detection Camera 123

List of Figures

1.1 Global Population Growth[15] 1

1.2 United Kingdom Wheat Production in T/Ha[21] 2

1.3 U.S. Corn Production in T/Ha[21] 3

1.4 Global Agricultural Land [21] 4

1.5 Food Prices and Related Riots in North Africa and the Middle East [13] 5

1.6 2008 Cereal Prices vs Food Riots[19] 6

1.7 2021 Global FPI[21] 7

1.8 Large Farm Equipment[12] 9

1.9 Precision Agriculture [5] 9

1.10 Fertilizer[26] 9

1.11 Plant Genomics[24] 9

1.12 U.S. Farmers, Farm Size and Total Land[25] 10

1.13 Historical Improvements to U.S. Corn Yields[16] 12

1.14 Fertilizing Rates[4] 12

1.15 Norman Boulaug Inspecting Wheat[3] 14

1.16 Multitrait Gene to Phenotype Model[8] 15

1.17 Crop Process Model[8] 16

1.18 Geophex Sensor[11] 18

1.19 Robot CAD Model..19

1.20 Geophex Gem-2 EM Sensor[11] 20

1.21 Geophex Sensor EM Data Heat Map[10] 21

1.22 Geophex Sensor used for Water Observation[20]................................ 22

1.23 Soil Moisture Equation 22
1.24 Crop Growth Causal Loop 23
1.25 Simple Wheat Models for Various Plots 24

2.1 Kansas State University Research Wheat Fields 26
2.2 Drone Imagery of the EPSCoR Field 27
2.3 Ground-level Perspective of Row Spacing in an EPSCoR Field 28
2.4 Robot Straddling the Wheat in a Field 29
2.5 Fender Assembly 32
2.6 Fender Fabricated 32
2.7 Robot CAD Model 33
2.8 Robot CAD Model Initial Build 34
2.9 Servo Mount CAD Model 35
2.10 Fabricated Servo Mount 36
2.11 Remote Fabrication Studio 37
2.12 Remote 3D Printing 38
2.13 Logic Level Shifter 39
2.14 Initial Electrical Box 39
2.15 PID Example Flowchart 41
2.16 Campus Tests 44
2.17 Gusset CAD Model 45
2.18 Gusset with Reinforcement CAD Model 46
2.19 Gussets Being Mounted for First Time 47
2.20 Mounted Solar Panels 48
2.21 Solar Power Generated Under Full Sun 49

2.22 Battery with Battery Protection Board 50
2.23 Battery Compartment 50
2.24 Initial Sensor Mount 51
2.25 Sensor Mount Redesign 52
2.26 Robot Electronics Schematic 54
2.27 GPS and Compass Antennae 56
2.28 Updated Electrical Box 57
2.29 IR Sensor 58
2.30 IR Sensor Schematic 58
2.31 TRIAD SPECTROSCOPY SENSORAS7265X 59
2.32 Adafruit VL53L0X Distance Sensor 59
2.33 Campus Tests 59
2.34 Visible Spectrum Sensor with Auxiliary Distance Detectors 60
2.35 Initial Field Testing 61
2.36 On Campus Drive Tests 62
2.37 Drive Tests at the North Fields 63
2.38 Drive Tests at Ashland Bottoms Wheat Field 64
2.39 Initial Sensor Data 65
2.40 Orange Robot in the Lab 67
2.41 Orange Robot at Ashland Bottoms 67
2.42 LattePanda Alpha 69
2.43 Topcon Developer Board 70
2.44 GPS Receiver 70
2.45 IR Sensor 71
2.46 IR Sensor Schematic 71
2.47 Visible Spectrum Sensor with Auxiliary Distance Detectors 72

2.48 OpenCV AI Camera 77

2.49 OpenCV AI Camera Mount 78

2.50 Power Meter 80

2.51 Solar Power Generated Under Full Sun 81

2.52 Robot Power Usage 82

2.53 Net Power of the Robot 83

2.54 Idle Power Use 84

2.55 3 Hour Power Consumption 86

2.56 Operating Voltage Levels 87

B.1 LabVIEW Communication Code 100

B.2 LabVIEW PID Code 101

1 LabVIEW Communication
Code 100

List of Tables

2.1 Battery Expectancy for 196.66 WH of Capacity 88

Introduction

The agronomy department at Kansas State University, through a project grant from the National Science foundation, EPSCoR award number 1826820, wants to retrieve more precise data to improved wheat modeling. The project will monitor different characteristics of the wheat and its environment, collect data, and use the data to design better wheat models. This paper discusses one method of data collection through the design of a robot that carries a bulk EMI sensor to gather moisture content. This data will then be included within the data to better model the wheat.

Chapter 1 discusses the global need for increased crop production through understanding of the various inputs and outputs of crop growth via better modeling of the crop as discussed in the sections on Genotyping and Phenotyping. These models can then be used to determine the best approaches based on the needs and requirements for future growth. Potential methods are discussed in the Agricultural Responses to Growing Food Demand and include more efficient farm equipment, more precise applications, fertilizer, and finding or developing new strands of crops with larger yields. Chapter 1 also discusses the proposed solution through the EPSCoR project, to gathering one aspect of the data required for the new models through monitoring the moisture content of the soil. It discusses the selection of the EMI sensor and the initial designs for a robot to carry the sensor through the field and record data. And it concludes in discussing the use case and relevancy of the data for future modeling.

Chapter 2 of this paper details implementation of the initial robot and sensor designs as proposed by the EPSCoR team. It begins by documenting the fabrication of the first prototype with respect to those initial designs. Chapter 2 then goes into detail documenting the required electrical design and the structural changes required for the robot to function as intended in the environment. It then discusses the mounting of the EMI sensor and the inclusion of the wavelength detection sensor designs. The field testing of the EMI sensor and the robot were then carried out. Additional camera sensors on the robot were proposed to explore AI and machine learning to control the robot's path planning are discussed towards the end of the chapter. The chapter concludes with test data about the robot's solar power generation and power usage for different scenarios within the environment to determine the robot's capability of meeting the requirements.

Chapter 3 then concludes this paper by discussing what requirements were met during fabrication of the robot and what work still is required to move the robot and EMI sensor portion of the EPSCoR project forward. It concludes that the hardware meets the requirements for the robot to traverse the environment but that calibration of the EMI sensor within the environment and more refined robotic software are necessary to retrieve the reliable moisture data required for the EPSCoR wheat modeling.

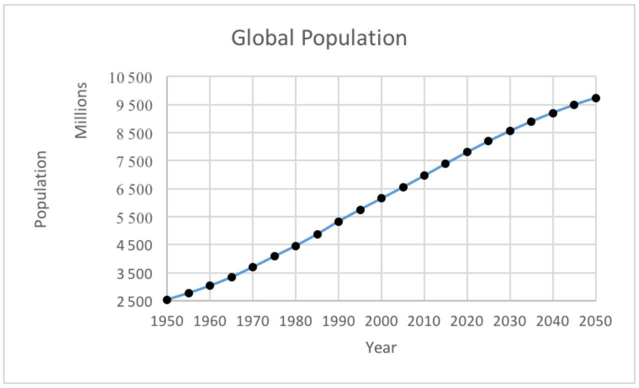
Chapter 1

1. Global Food Needs for 2050

The United Nations published a report expecting the population of the world to increase to 9.7 billion people by 2050[15]. And while the global growth rate has been slowing since the early 1970s (see Figure 1.1), the population is already stressing the global food supply and is outpacing the trajectory of food production over the same time period.

Figure 1.1

Global Population Growth



Note. Although slowing, the exponential growth rate of the world's population is considerable. [15]

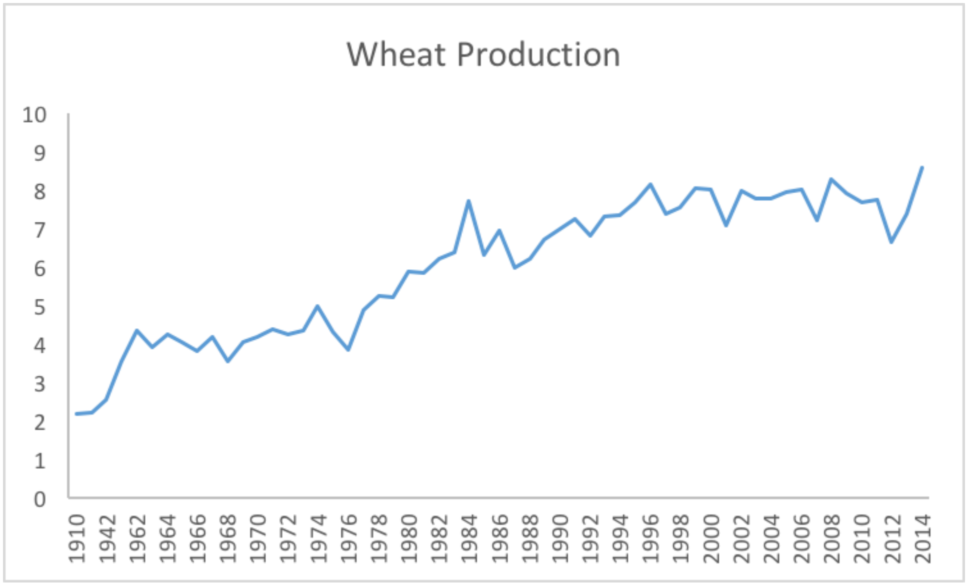
With the population of the Earth continuing to rise, the amount of food required to feed the world also is growing respectively. However, current technological improvements are not keeping pace with the anticipated increase of global food production.

1.1 Globalization's Impact on Arable Land

According to the 2006 Food and Agriculture Organization (FAO) report[2], the conservative estimate for 2050 food production needed is 50% to 70% higher than current food production. This increase represents a compounding annual growth rate of around 1.75%, nearly double the predicted increase of food production with the current technology.

Figure 1.2

United Kingdom Wheat Production in T/Ha

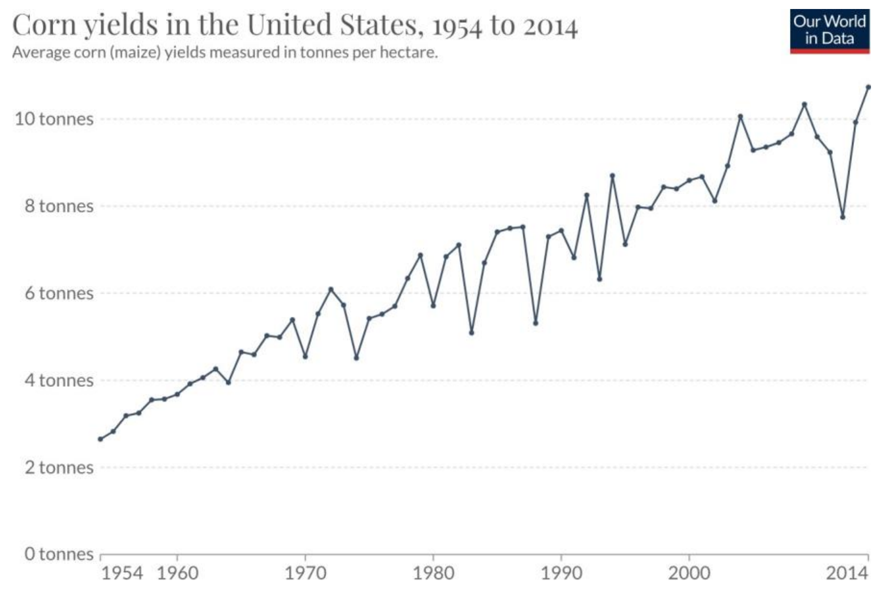


Note. Early 21st century technological advances led to increased wheat production. However, as all available farmlands became planted, that rapid growth has been difficult to maintain and has almost approached stagnation, or zero growth, in the past decade. [21]

Globalization in the 21st century has uncovered all available land. To understand the implications of this on the future of agriculture, look at the United Kingdom: a first world country that has been on the forefront of agricultural technology and yet has had all its land discovered for centuries. The United Kingdom has some of the most advanced farming technology that could be used to increase crop per acre. In the 20th century, larger farm equipment was used to produce crops more efficiently; however, the same rate of growth has become much more difficult to achieve. Figure 1.2 shows that the overall production of wheat has increased dramatically in the past 100 years, but the growth rate has slowed dramatically in the past few decades, recently nearing stagnation—a zero percent growth rate—for the main cereal crop.

Figure 1.3.

U.S. Corn Production in T/Ha

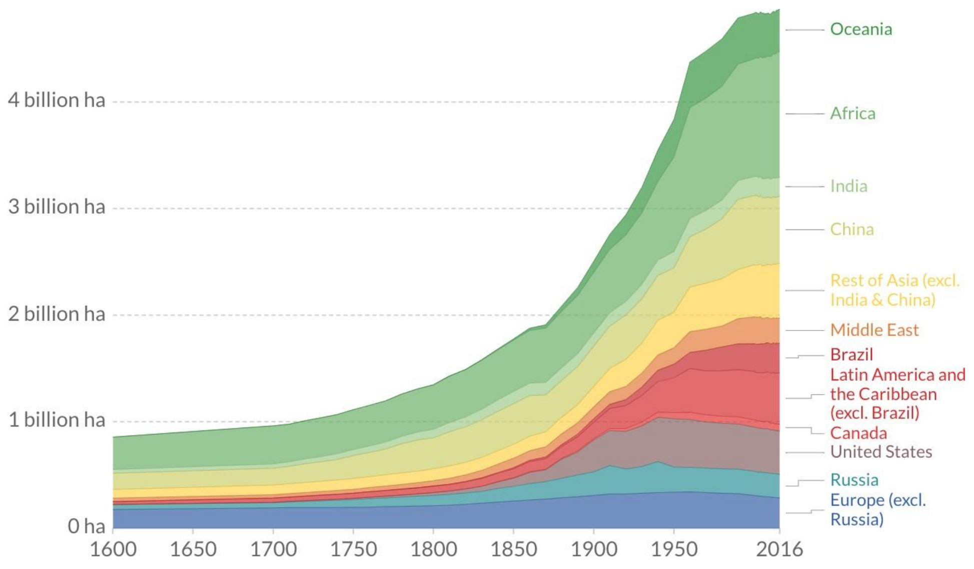


Note. U.S. corn production saw growth after World War II; however, land saturation is now leading to a slower growth rate. [21]

Figure 1.3 shows that the United States is following a similar trend with corn production. The 1900s saw tremendous growth after World War II. The increase in fertilizer use along with the hybridization of crops that had previously been cross-pollinated led to a crop production boom. However, U.S. land is reaching a similar saturation point to the United Kingdom, and the same increase to inputs are realizing less growth year over year.

Figure 1.4

Global Agricultural Land



Note. For certain regions, exploration led to more available land suitable for agricultural purposes. [21]

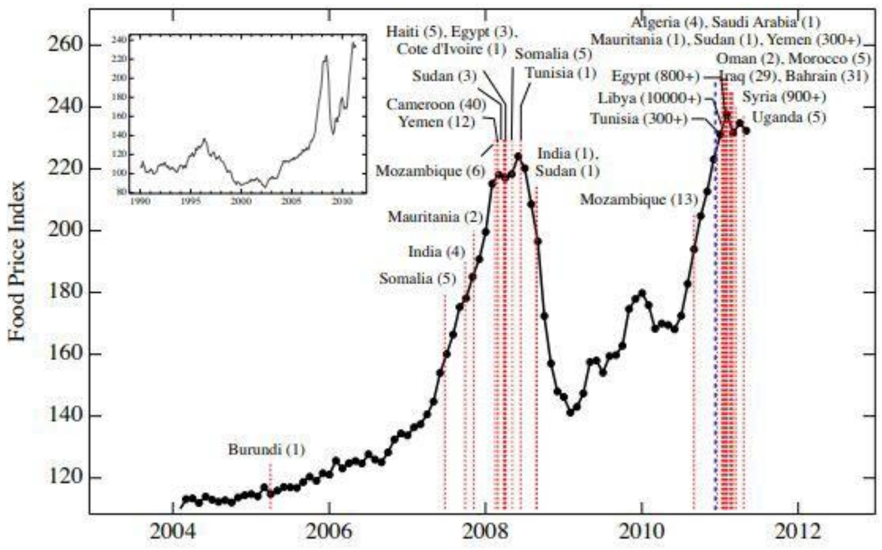
The world's production of crops per acre continues to grow at a compounding growth rate of 3%, similar to the United Kingdom before 1990. But with total arable land now stagnate, as shown in Figure 1.4, it is more important than ever that the world's production per acre does not stall, as has been seen of late in the United Kingdom. As shown, the amount of usable land in Europe has remained nearly constant since 1600, while the land increases during global exploration from 1600 until 2000 only recently have leveled off. The United Kingdom's agricultural stagnation has already taken place, and the current trajectory is following suit for the rest of the world about 100 years later. To grow enough food for a continuously growing population in a world with a constant amount of land, food production will need to increase per unit of land. Work to continue to grow more food per acre has four main approaches: larger farm equipment, better fertilizer applications, precision agriculture, and plant genomics.

1.2 Effects of Food Shortage and Insecurity

Food shortages historically have led to malnourishment and even conflict. Studies by the FAO[7] [6] show that a below average intake of Kcal/day has a strong correlation to poverty. And increases in food prices lead to an increase in conflict in a region, which usually leads to another increase in food prices.

Figure 1.5

Food Prices and Related Riots in North Africa and the Middle East



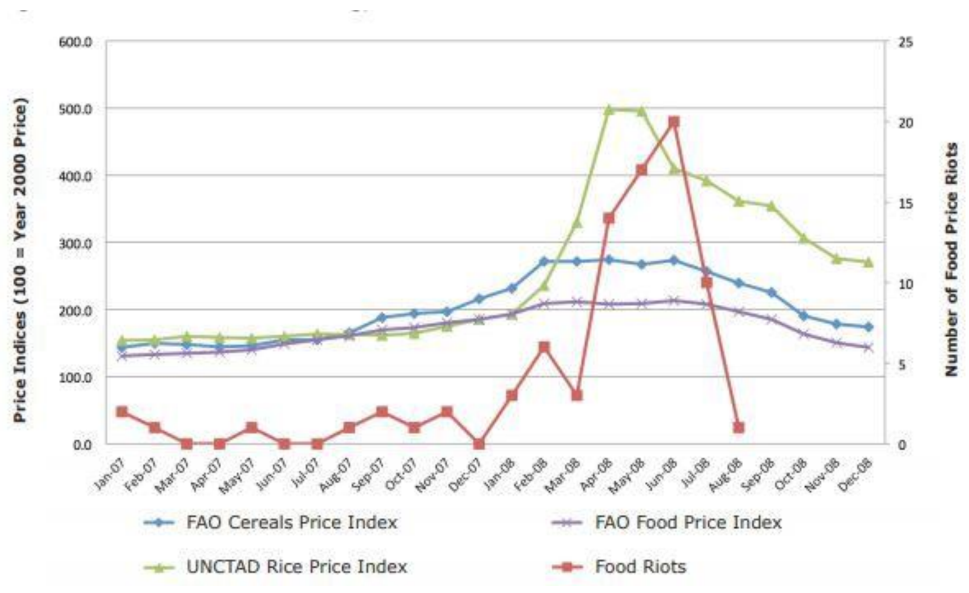
Note. Each country experienced a food riot related to increasing food costs. The numbers in parentheses represent the number of individuals killed in each riot. [13]

Figure 1.5 shows the correlation between the rising prices of food in North Africa and the Middle East and the number of food-related riots in each region. As prices began to rise because of the 2008 financial crisis, food-related riots increased from near zero to 13 related incidences, resulting in more than 80 deaths. Another spike in food prices came in 2011 and resulted in nearly 13,000 related deaths. While North Africa and the Middle East are known for their conflicts, the same correlation is noted by the FAO regarding global trends.

By increasing food production, adding to the supply, prices would have downward pressure across the market, leading to lower prices and higher food security.[6] The result would be fewer conflicts and continued population growth. In Figure 1.6, the same correlation is shown. In 2008, when the global food price index rose dramatically, especially in respect to rice, conflicts rose respectfully.

Figure 1.6

2008 Cereal Prices versus Food Riots

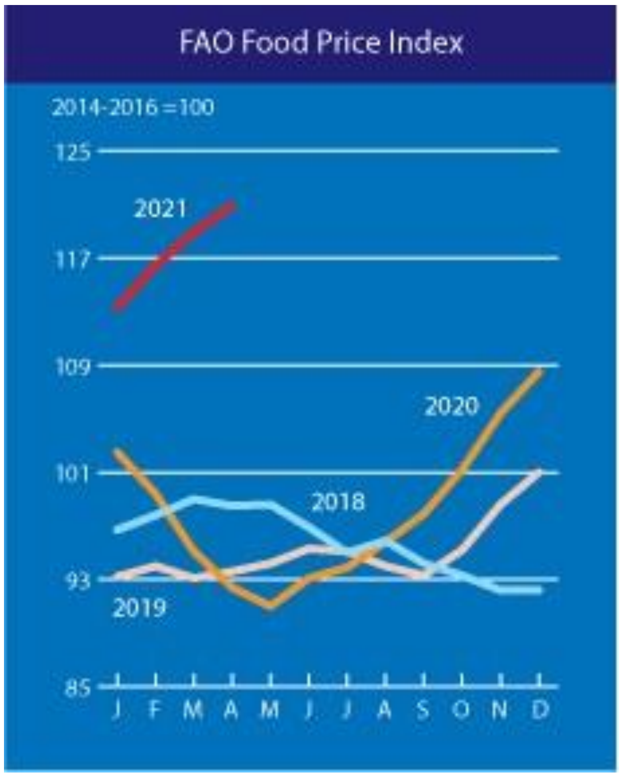


Note. As the price of cereal grain increased in 2007 and 2008, the incidence of food-related riots also increased.[19]

As we study the effects the financial crisis had on food and conflict, the COVID-19 global pandemic presents another time frame for review. The pandemic has caused an increase in the food price index of more than 25% (see Figure 1.7).

Figure 1.7

2021 Global FPI[21]



Though it is difficult to ascertain whether food prices cause conflicts or if conflicts cause food shortages, research suggests that the flywheel effect is in play. As prices increase, conflicts increase, which in turn makes prices increase[19]. Hindsight will show if the conflicts that are occurring have been influenced by food shortages, or the reciprocal, or both, but the first quarter of 2021 has brought many new international conflicts resulting in thousands of deaths and many more who are displaced. If crop production slows as the world's population grows, increased conflict is a likely result, especially with increasing urbanization. Any disruption to a fragile global food system will lead to larger food-related conflicts. It is important, now more than ever, to make agriculture more robust. If food production can grow more rapidly, the number of people inhabiting the Earth can increase and the number of conflicts related to food security can be dramatically reduced.

1.3 Production per Acre

With nearly all available agricultural land discovered, the effects of large farm equipment on production diminishes year over year. In the 21st century, production of staple crops began to plateau[21]. With new arable land being more difficult to find (see Figure 1.4), large tractors and combines are no longer creating the advantage they once had. While they have allowed farmers to greatly increase the amount of land that can effectively be cultivated, what matters most now is how efficiently one can produce per acre.

In recent years, there have been multiple different new approaches to increase unit area production. There has been work in mitigating crop loss—more precisely planting, managing, and harvesting. More attention is being given to the individual plant, more densely packed crops, crops that are planted in vertical rows, and more. Each method attempts to add to the portion of the pie that increases the total production of the world's food to the 2050 goal of 50% to 70% increase. Separately their increases barely move the needle, but together, each incremental percentage of crop planted, harvested, and saved compounds into something much greater and grander than the individual farmer feeding his family and neighborhood.

1.4 Agricultural Responses to Growing Food Demand

There are many approaches to increasing food production. Some include building larger, more efficient farm equipment; using more precise sensing and application techniques; devising better fertilizer applications; and selecting better crop genes (see Figures 1.8 to 1.11). Each technique plays a part in the larger picture: increasing food production.

The Four Main Approaches

Figure 1.8: Large Farm Equipment[12]

Figure 1.9: Precision Agriculture [5]

Figure 1.10: Fertilizer [26]

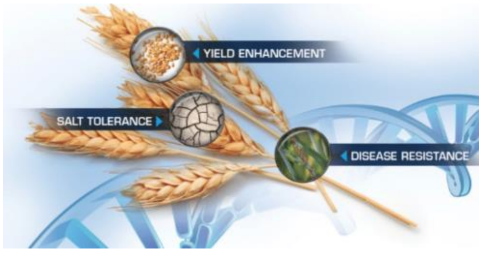


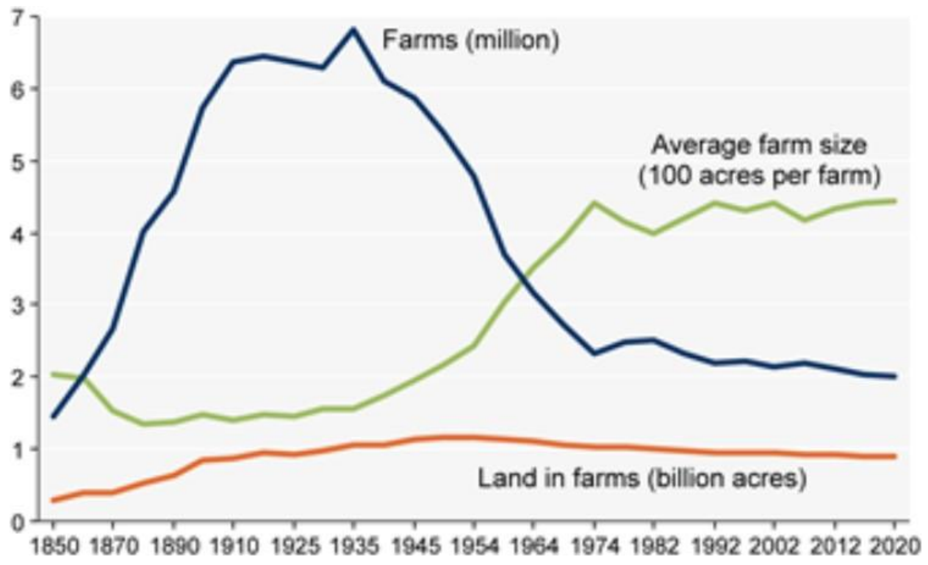
Figure 1.11: Plant Genomics [24]

1.4.1 Large Farm Equipment

In the 20th century, farmers greatly increased production by investing in larger, more efficient farming equipment. And large farm equipment will continue to improve and have a role in 21st century farming, especially by reducing the number of farmers per acre and mitigating skilled labor shortages. However, the United Kingdom's production stagnation proves that while large farm equipment allows more land to be farmed, this equipment alone cannot make the land to produce more crops per area.

Figure 1.12

U.S. Farmers, Farm Size, and Total Land



Note. While the amount of farmland has changed little in the past century, the number of farms has decreased and the size of each farm saw expansion before finally leveling off. [25]

Indeed, Figure 1.12 shows that as the size of tractors grew tremendously in the 1900s, the effect this had on production became limited after 1950. From 1950 until about 1974 ,the farm size continued to grow, but fewer farmers are now farming larger sections of land. This trend is expected to continue as total farmland is reduced by urban expansion.

1.4.2 Precision Agriculture

A more recent approach of the past decade has been in precision agriculture. An estimated 10% to 40% of global crops are lost because of pests and diseases.[23] By more precisely incorporating countermeasures, annual crop loss can be reduced. Research in this area has many facets, from drone imagery monitoring crop health, to scouting robots and soil sensors. The goal of each approach is to increase the data and knowledge for each individual plant. Not only does this type of approach have the capacity to increase production significantly,[6] it can also reduce overhead costs associated with pesticides and water management systems.

Unmanned Aerial Vehicles (UAVs) scan the target field, sensing different characteristics of crop health. The main consideration is crop stress through temperature and image processing techniques. By knowing which crops are stressed, the farmer can know how to effectively respond to the crop's needs (e.g., spot irrigation). By specifically watering the crops that are stressed, the water intake the crop gets can alleviate the crop stress, potentially increasing production and mitigating crop loss.

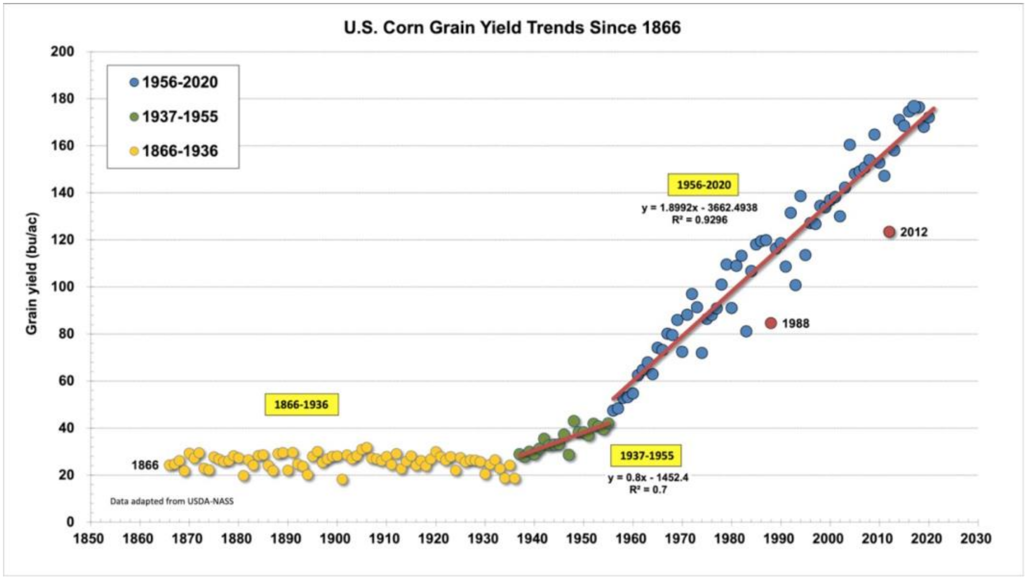
Scouting drone robots are similar to UAVs in that their main objective is to observe the field. However, the main advantage is the difference in perspective. While UAVs have the benefits of ease, unmanned ground vehicles (UGVs) can explore beneath the canopy and up close to the crop. From this vantage, the crop can more closely be inspected for damage from pests or various diseases. UGVs can also be used to sense different characteristics of the plant or the soil that require closer contact to the target. Soil moisture sensors, among others, can detect changing characteristics of the soil throughout the year. By incorporating more data into the planting, managing, and harvesting of the crop, the farmer can learn better techniques and responses to care for the crop.

1.4.3 Fertilizing

The advent of nitrogen fertilizers and various pest controls in the early 1950s carried and expanded the advancements from World War II until then. In the 1940s and early 1950s, the annual growth of production was 0.9 bu/acre of corn; it doubled after the 1950s with the inclusion of new inputs like fertilizer.[16]

Figure 1.13

Historical Improvements to U.S. Corn Yields

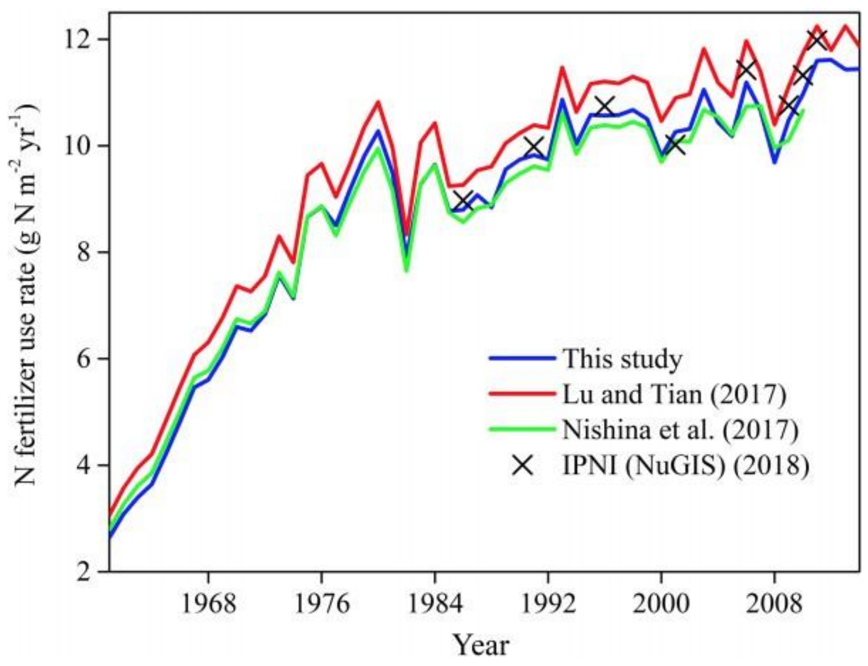


Note. The introduction of tools, such as fertilizer, resulted in greater corn yields.[16]

Figure 1.13 communicates two substantial changes to the yield of corn in the United States. The latter change due in large part to fertilizer seems to continue off the chart; however, fertilizer applications are reaching their current limitations shown in Figure 1.14.

Figure 1.14

Fertilizing Rates



Note. The use of fertilizer appears to have neared its limit in the past few decades. [4]

The result of saturating fertilizer reaching its limit is shown in more recent corn yield trajectories (see Figure 1.3) that seem to have slowing growth. The trend is expected to be similar for other cereal crops as technologies, inputs, and genomics continue to be researched. Wheat is experiencing the effects of fertilizer saturation rate much more heavily (see Figure 1.2), limiting future wheat production growth rates.

1.4.4 Genotyping

The conventional techniques led to a doubling of crop production between 1960 and 2015. However, the growth rate of production from these traditional methods used in the green revolution is slowing and currently is between 0.9% and 1.6%, compared to the projected need of 2.4% annual demand.[18]

The approach that has had the largest potential for the 21st century is intelligent plant phenotyping. With more data being collected for each crop, scientists have been able to more accurately describe the characteristics of a different breed of a particular plant. By growing multiple different strains and characterizing their genes, farmers can plant

variants of a crop that perform better in their region. However, the main challenge with utilizing this approach effectively is the large amount of data that is required to accurately map the plant's traits and describe its environment in equations to develop models.
This paper discusses one such approach using a combination of sensors and robotics to phenotype different plots of research wheat. The goal is to be one portion of the immense data collection required to model the plants to ultimately increase crop production within an acre.

1.5 Wheat Phenotyping

More than half of the world's caloric intake is in staple crops, accounting for 51% of the world's diet.[17] Wheat is a significant portion of the cereal crops used, and it is mainly consumed in North America and Europe, contributing to one third of the world's diets. [9] In total, wheat consists of about 15% of the global caloric intake. [17] Any increase in wheat production will significantly add to the global net food supply.

Figure 1.15

Norman Boulaug Inspecting Wheat

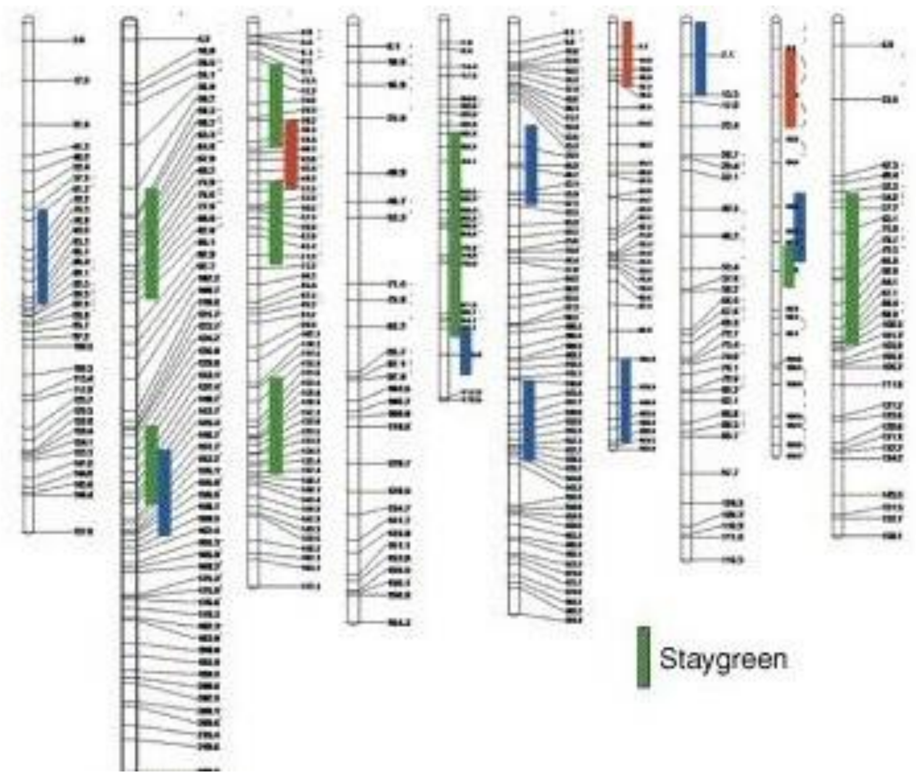
Note. Norman Boulaug is famous for wheat phenotyping.[3]

A famous example of wheat phenotyping is Norman E Borlaug, [3] who would impact the growth of wheat production by nearly threefold per unit acre by carefully monitoring specific traits, depth of soil, and seed spacing. The largest part of the project was to use dwarf spring wheat that had shorter stalks and larger heads. These shorter stalks were able to carry the larger heads without falling over or "lodging". These traits allowed for wheat production to increase greatly and lead to a green revolution in wheat production. Today, the project continues as wheat production per acre must continue to grow to match population growth.

The main method for increasing wheat production in this paper is the implementation of phenotyping through statistical and remote sensing methods. Using data that characterizes a plant's growth, a prediction model can be created that extrapolates different characteristics of the individual wheat plots. Within the same type of wheat, different generations may have pronounced traits—similar to how a survival of the fittest model for evolution breeds larger, faster, stronger animals—breeding more generations leads to new possibilities in wheat yields and diversity. This is especially important as worldwide "agriculture has shifted to monoculture, resulting is a significant reduction in plant diversity."[18] Using seed breeding that models the diversity to search for genes with higher yield that can handle drought situations artificially can mimic biodiversity in a monoculture application.

Figure 1.16

Multitrait Gene to Phenotype Model

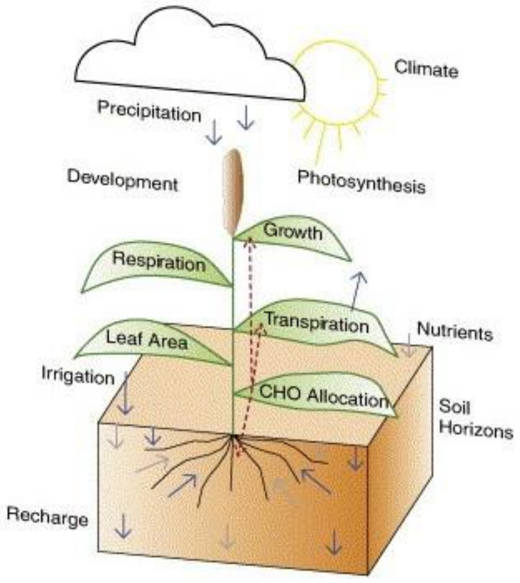


Note. A map showing part of a genome that is associated with the Staygreen trait [8]

By understanding the types of traits present in a wheat genome, a more targeted seed approach could increase wheat in their respective regions. Historically, a trait that is shorter produces more yield because it takes less time for the plant to reach maturity and production is not based on the height of the plant but the size of the head. It will also require less water over the lifespan of the crop because of its smaller stature.

Figure 1.17

Crop Process Model



Note. Crop growth is a multistep process dependent on many factors. [8]

Some traits that would be beneficial are shorter height, a larger head, less water requirement, and resistance to pests. Each of these types of wheat will then be planted in regions that would benefit most from their traits. For example, a wheat trait that fares well with less water could be implemented in areas like western Kansas and Colorado, whereas traits that increase production in proportion to water might be better suited to regions with more rainfall or access to irrigation, such as Iowa. Traits can also help identify wheat types that are more resistant to disease, pests, and a host of other useful qualities that improve wheat production overall.

To predict different characterizations of a wheat plot, a dynamic crop growth models (CGMs) has been created that models' wheat's growth cycle.

$$CGM = F\,(Y_{T\,LN\,i}, Y_{SRE}, Y_{AM\,i}, \Omega_i) \tag{1.1}$$

Where:

$$Y_{T\,LN\,i} = U_{T\,LN} + Z_i U_{T\,LN} \tag{1.2}$$

$$Y_{AM\,i} = U_{AM} + Z_i U_{AM} \tag{1.3}$$

$$Y_{SREi} = U_{SRE} + Z_i U_{SRE} \tag{1.4}$$

$$Y_{MTUi} = U_{MTU} + Z_i U_{MTU} \tag{1.5}$$

And:

- YTLNi etc. are the values of the physiological traits observed for the ith genotype
- Weather and management data of environment k are represented as Ω k.

Just like nearly all models, the equation represented here is a simplification of the many things that take place for a wheat plant to grow well. The main inputs are classified as sunlight radiation, water intake, temperature, growth constants, and leaf area index. The more accurately these inputs are measured or determined, the more accurate the model will be at predicting the growth trend of a wheat plot. In addition, any new characteristics that affect the plant's growth cycle, such as soil characteristics, will make the model more accurate.

The more accurate the model created is, the faster the research timeline would be for new crop types. And specific crops can be placed in their respective regions more accurately. To do this, a sensor is used in the field to measure the different inputs that the equation requires.

1.6 Wheat Sensor Integration

To accurately determine the influence of the inputs on the model, precise and consistent sensors need to be created. These sensors will monitor the plant's growth over its life cycle. The sensor needs to be robust, accurate, precise, and able to measure throughout the season. The more data that is collected on the plants, the more accurately the model can describe what the plants are experiencing.

Figure 1.18

Geophex Sensor

Note. Saquib walking the Geophex Sensor in the field to record test EM data

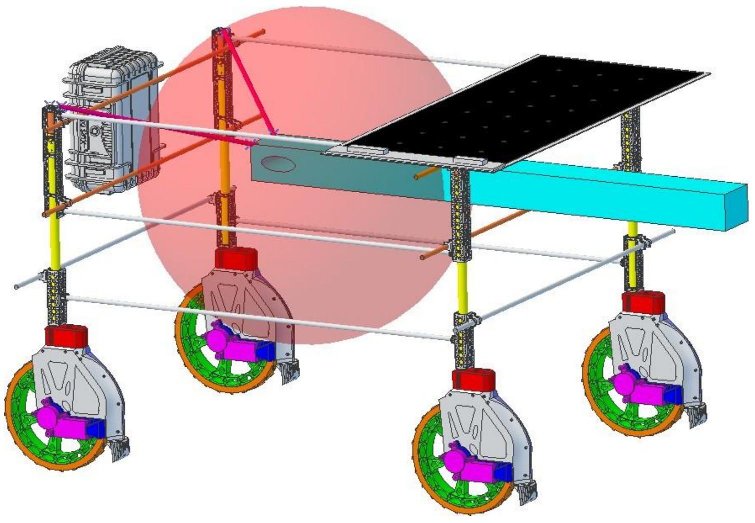
The agronomy department at Kansas State University wants to know what role continuous bulk monitoring of environmental characteristics of wheat growth can play into learning specific wheat trait patterns. Specifically, the sensor designed will measure water moisture of the soil throughout the plant's life cycle carried by a robot designed in Figure 1.19 and compared to test data gathered by walking shown in Figure 1.18. By using the Fourier transformation on the signal, more than just water moisture may be able to be detected. By knowing the patterns of moisture, nutrients, disease, and pests involved in a particular wheat plant's growth cycle, a statistical model can be more accurate in analyzing or even predicting specific wheat traits that are desirable.

1.7 Robotic Autonomous Sensing

To create a more accurate statistical model, a large quantity of data is required. To accomplish greater and more consistent measurements, a robotic rover design was conceived to carry the electromagnetic (EM) sensor to measure the conductivity of the soil at specific frequency intervals (see Figure 1.19).

Figure 1.19

Robot CAD Model



Note. A robotic rover can carry the electromagnetic sensor to collect data to analyze crop production.

The sensor can measure area moisture without penetrating the ground. The data that is obtained is then inverted and run through a Fourier transformation to extract the usable data. By having a robot carry the sensor through the field, more consistent and precise measurements can be made with less human intervention and labor hours. However, the main limitation to using an in-field robot is the sensor's sensitivity to metal. The composite design of the robot attempts to limit the sensor's exposure to metallic interference of the signal by maintaining a bubble around the sensor with no metal. Though this limits the effects, it is near impossible to eliminate it.

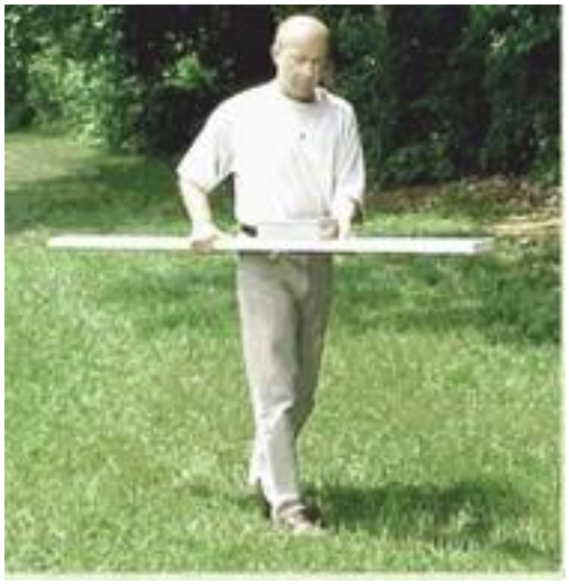
1.8 Geophex EM Sensor

The robot is designed around the Geophex Electromagnetic (EM) sensor it carries through the field. The sensor has three coils that carry an electric current at multiple frequencies determined by the user. The magnetic field is measured as it leaves the coils and arcs into the ground. The influence the surroundings have on the sensor are measured from the third
coil after the field has traversed the subsurface. [11] Material that has a higher conductivity

returns a bigger response. A change in the sensor's frequency alters the depth of observation, with higher frequencies measuring a bulk volume closer to the surface.

Figure 1.20

Geophex Gem-2 EM Sensor

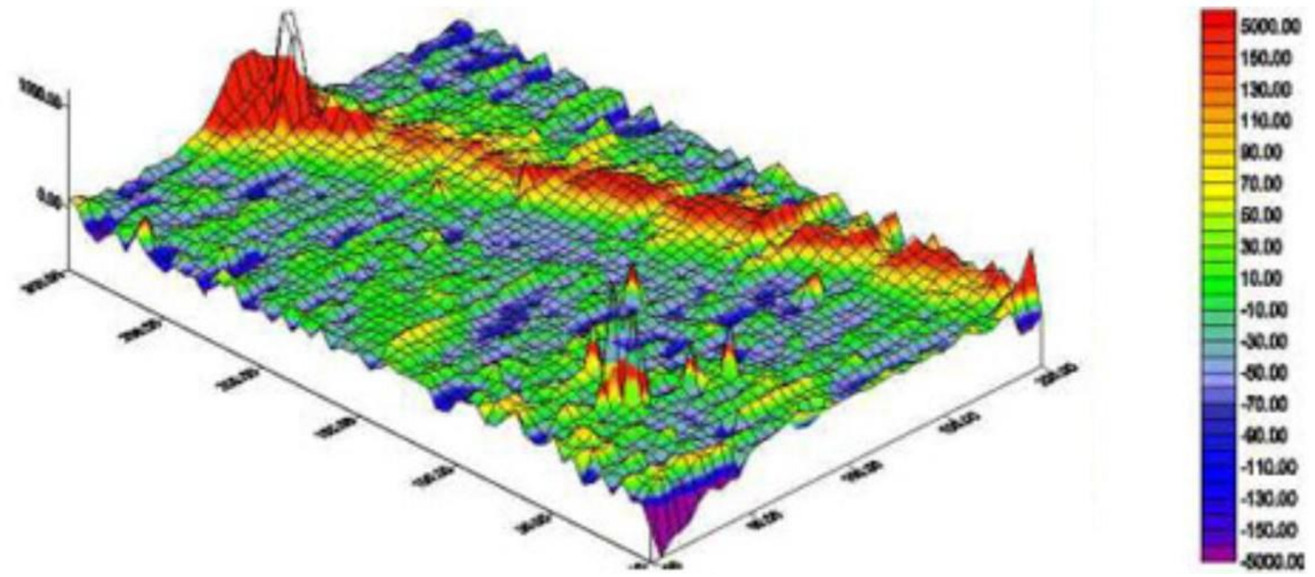


Note. The Geophex Gem-2 EM Sensor records data at the field level.[11]

The EM sensor's capabilities are primarily used to observe and record subsurface structures and objects. By selecting different frequencies, different depths are recorded and stitched together to create a heat map of the underground terrain.[10]

Figure 1.21

Geophex Sensor EM Data Heat Map

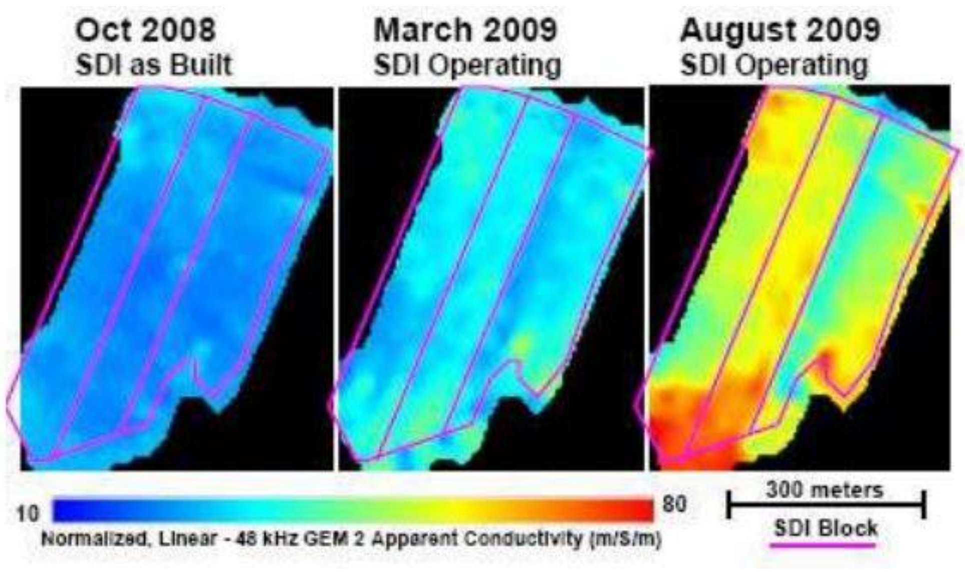


Note. Geophex data collected at different frequencies results in a pipe being found displayed in red on the heat map[10]

The Kansas State University Phenotyping Project will determine if EM sensor measurements can be used to determine other characteristics of the surface and the subsurface terrain of the Earth similar to the study shown in Figure 1.21. By measuring at multiple different frequencies and over the course of multiple months, the project will see if the sensor can be used to measure moisture content of the soil near the surface and use Fourier transformations to characterize nutrients within the soil.

Figure 1.22

Geophex Sensor Used for Water Observation



Note. Geophex Sensor detecting electrical conductivity changes water caused in the soil[20]

While different frequencies are usually used to produce different depths of analysis, it could theoretically be possible to use these transformations for other purposes.[20]

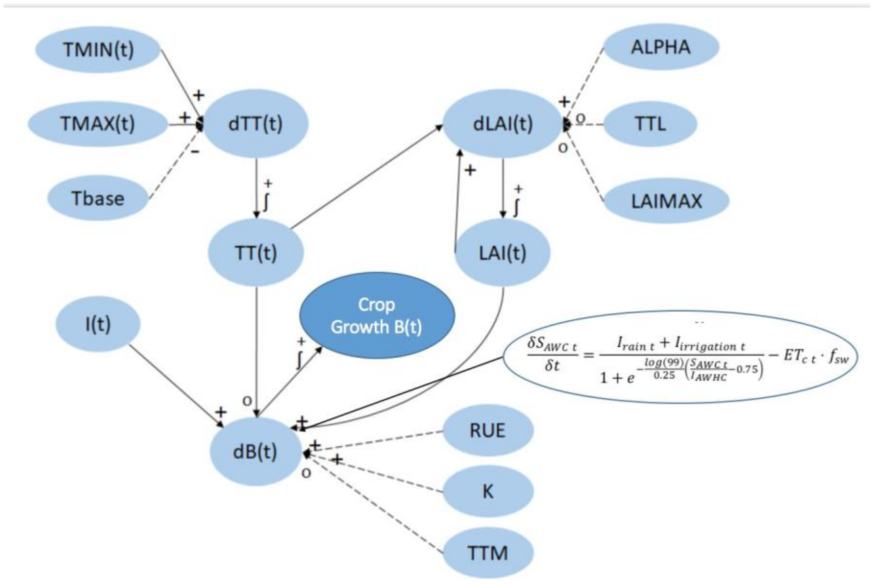
The resulting expected data output of water moisture and nutrient resources within the soil is to then be used as inputs into phenotyping models. The wheat traits that produce better results from field trials and models will be selected for farm scale production. Each research plot in the field will be characterized by the inputs from the soil sensor, as well as weather data and other metrics. The model is then generated to display the expected growth of each wheat plot throughout the season. These models are updated to reflect the accuracy of the resulting seasonal harvest of the wheat. See Figure 1.23 for an equation being developed to explain the relationship between soil moisture and wheat growth:

Figure 1.23

Soil Moisture Equation

$$\frac{\delta S_{AWC\,t}}{\delta t} = \frac{I_{rain\,t} + I_{irrigation\,t}}{1 + e^{-\frac{log(99)}{0.25}\left(\frac{S_{AWC\,t}}{I_{AWHC}} - 0.75\right)}} - ET_{c\,t} \cdot f_{sw}$$

This equation is then added to the causal loop in conjunction with the simpler descriptive variables such as biomass, leaf area index, and time temperature to affect the overall model of the plant's growth (see Figure 1.24)



TMIN(t)
TMAX(t)
Tbase
dTT(t)
TT(t)
dLAI(t)
LAI(t)
ALPHA
TTL
LAIMAX
Crop Growth B(t)
I(t)
dB(t)
$\frac{\delta S_{AWC\,t}}{\delta t} = \frac{I_{rain\,t} + I_{irrigation\,t}}{1 + e^{-\frac{log(99)}{0.25}\left(\frac{S_{AWC\,t}}{I_{AWHC}} - 0.75\right)}} - ET_{c\,t} \cdot f_{sw}$
RUE
K
TTM

Figure 1.24

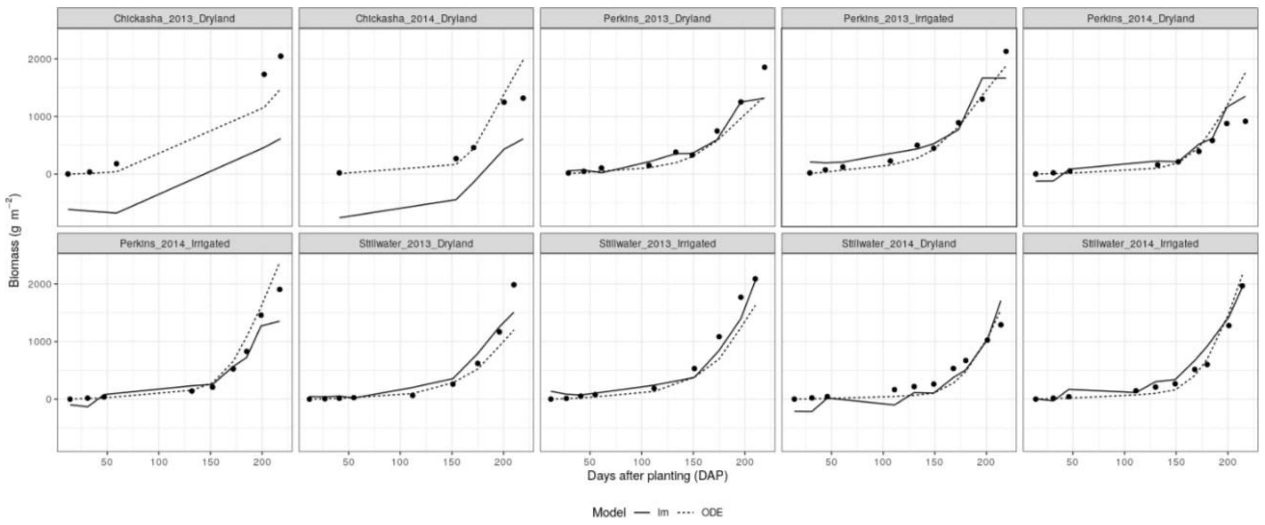
Crop Growth Casual Loop

Note. This example shows how the causal loop might be affected by the increase in data.

The result would be that the value of the change in biomass of the plant represented by dB(t) would more accurately describe the plant's growth cycle. A model would then be generated for each of the more than 200 research plots to describe each phenotype, with the hope of predicting the wheat with the best biomass return.

Figure 1.25

Simple Wheat Models for Various Plots



Note. Data is recorded to determine the conditions right for producing wheat with the highest biomass.

Figure 1.25 shows how this process has been carried out during the project using just the simple wheat model. With the additional information from the soil moisture and potentially the nutrients as well, the model can more accurately describe the plots. The more accurately they are described, the better precision the models can predict wheat phenotypes that produce more biomass. To accurately and reliably retrieve this data from the soil, the bulk EM sensor and the unmanned ground vehicle were designed and tested.

1.9 The EPSCoR Project

Chapter 1 has discussed the need for increased understanding of the various inputs and outputs of crop growth through better modeling of the crop as discussed in the sections on genotyping and phenotyping. These models can then be used to determine the best approaches based on the needs and requirements for future growth. Potential methods were discussed in the Agricultural Responses to Growing Food Demand section and include more efficient farm equipment, more precise applications, fertilizer, and finding or developing new strands of crops with larger yields.

The agronomy department at Kansas State University, through a project grant from the National Science Foundation in the United States via the Established Program to Stimulate Competitive Research (EPSCoR), wants to know what role continuous bulk monitoring of environmental characteristics of wheat can play into modeling yields. The project will monitor different characteristics of the wheat and its environment, collect data, and use the data to design better wheat models.

Chapter 1 also discusses the proposed solution through the EPSCoR project: gathering data required through monitoring the moisture content of the soil. It discusses the selection of the EMI sensor and the initial designs for a robot to carry the sensor through the field and record data. And it concludes in discussing the use case and relevancy of the data for future modeling.

Chapter 2 of this paper details the implementation of the initial robot and sensor designs as proposed by the EPSCoR team. It begins by documenting the fabrication of the first prototype with respect to those initial designs. Chapter 2 then details the required electrical design and the structural changes required of the robot frame from the initial design for the robot to function as intended in the environment. It then discusses the mounting of the EMI sensor chosen by Jonah Smith and Dr. Kulesza. the addition of the wavelength detection sensors from Jordan McClellan at Langston University, and the redesigned sensor by Ben Weinhold and Calvin Dahms. The field testing of the robot was then carried out by Calvin Dahms and with the testing design structure created by Saquib () with EMI data collected from the sensor. Additional robot sensors proposed by Calvin Dahms and Sujith Guturu to explore AI and machine learning to control the robot's path planning are discussed toward the end of the chapter. The chapter concludes with test data about the robot's solar power generation and power usage for different scenarios within the environment to determine the robot's capability of meeting the requirements.

Chapter 3 then concludes this paper by discussing the future.

Chapter 2

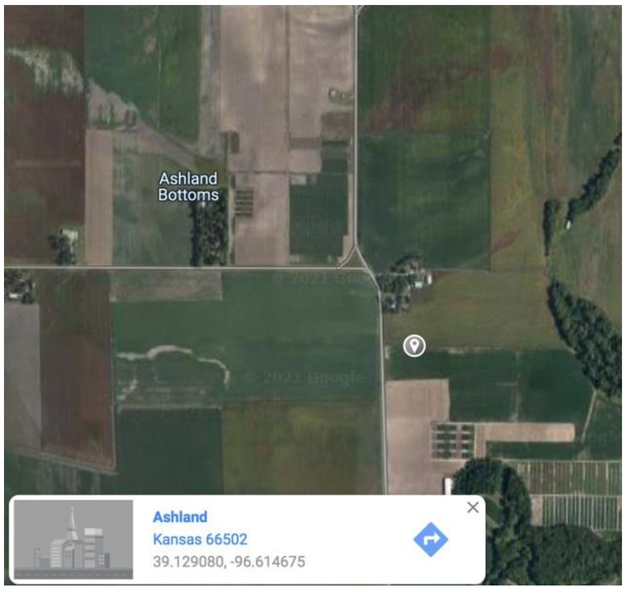
Robot Design

2.1 The Field Environment

The Unmanned Ground Vehicle developed by the Kansas State Biological and Agricultural Engineering department must be designed to the constraints of the field it is meant to operate in, as well as the deliverables for carrying the sensor. The research wheat fields that the robot is designed around are located south of Manhattan, Kansas, in Ashland Bottoms. The field contains 360 small plots of wheat specifically designed to be uniform. The width of the plots are 135 cm, a length of 200 cm, a dynamic height throughout the growing season of up to 100 cm, and a spacing between rows of 31 cm (see Figures 2.1 to 2.3).

Figure 2.1

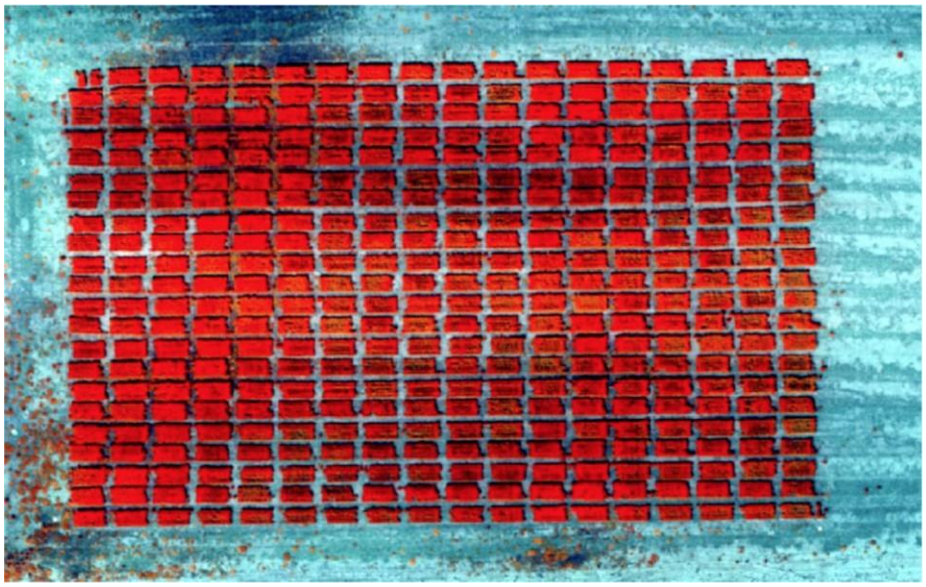
Kansas State University Research Wheat Fields



Note. The fields, south of Manhattan, Kansas, contain 360 small plots used to study wheat.

Figure 2.2

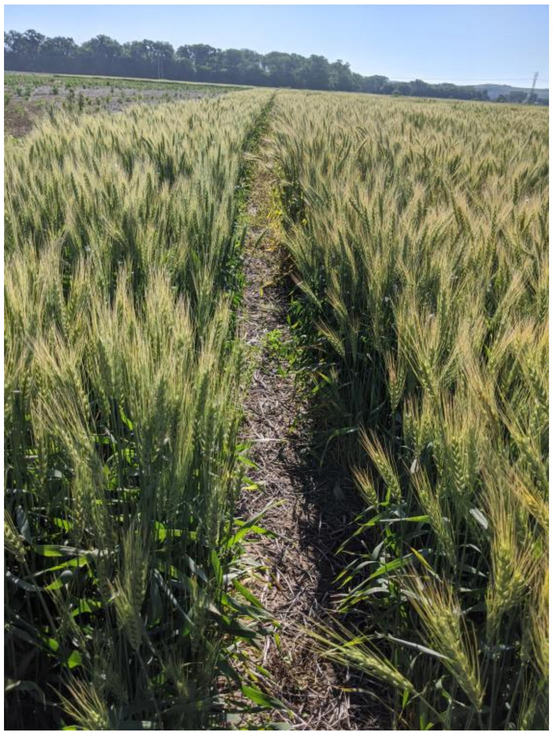
Drone Imagery of the EPSCoR Field



Note. An infrared drone image depicts layout of Kansas State University research field.

Figure 2.3

Ground-level Perspective of Row Spacing in an EPSCoR Field

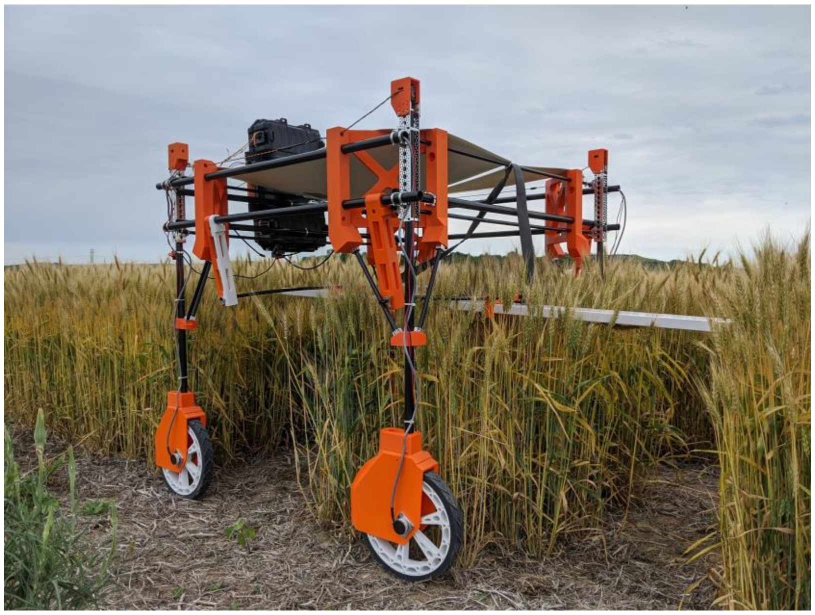


Note. Kansas State University research fields shows evenly spaced rows, a feature key to crop study.

The images show near uniformity of the field plots, with slight variations of plot dimensions and locations. The robot must be designed to be able to traverse through the field without interfering with the wheat. The wheels must fit between the rows and the sensor must not run into the wall of wheat on either side of the row. To satisfy these criteria, it was determined that the robot would have wheels on both sides of a particular wheat plot, straddling the wheat. The robot's width would adjust manually to account for slight variations in plot dimensions. Figure 2.4 shows a representation of a robot.

Figure 2.4

Robot Straddling the Wheat in a Field



2.2 Initial Proposal

The Biological and Agricultural Engineering Department at Kansas State University is developing an autonomous vehicle to carry the EM sensor for soil characteristic measurements.

This vehicle is designed of 1.27 cm diameter carbon fiber tubes, metal C channels and clamps for attachment, and 3D-printed ABS plastic for components such as the fenders and wheels. The robot is 1.5 m long and has a manually configurable width of 0.6 m to 1.5 m. Each wheel is designed to turn independently using motors mounted on the shoulders of the robot to increase the robot's mobility in the field. The robot can therefore turn its wheels in any direction, allowing the robot to move forward or slide left or right without changing the orientation of the robot. Each wheel is made of 3D-printed ABS plastic that is designed to be small enough to traverse between the rows of wheat plots without interfering with the crop, while being capable enough to traverse the rugged terrain. The wheel will do so by incorporating TPU plastic tread on the outer portion of the wheel to grip the ground and provide traction.

Originally, the robot was designed to carry a custom EM sensor built by Jonah Smith, an undergraduate student at Kansas State University. The sensor would consist of two plastic housings designed by Calvin Dahms that would be wrapped with copper coiling to be energized, creating an electric field. By adjusting the distance between the coils and the frequency of the current, the receiver coil would be able to record different depths of measurement and frequency of response, allowing for robust electrical conductivity data of the soil to be captured.

However, the design, build, and calibration timeline of the sensor does not fit within the time frame of the overall research project with the wheat and was set aside for an off-the-shelf EM sensor from Geophex. The GEM2 sensor is a 1.67 m board with three coils in a fiberglass housing that Geophex has calibrated already. It comes with an out-of-the-box receiver from Trimble and software for recording and interpreting the data. In addition, the sensor can be read by serial communication or via Bluetooth connection, allowing versatility in the robotic design. Purchasing the Geophex GEM2 EM sensor allows the research time to be spent on the construction of the vehicle and data collection instead of developing and calibrating a redesign of a sensor currently available on the market. The Geophex EM sensor will map the bulk apparent conductivity of the soil. While the multifrequency capabilities of the sensor account for different depth of investigation of the subsurface, the project intends to model the data at different frequencies to determine if other soil properties can be ascertained.

2.3 Environment Around the Sensor

The EM sensor from Geophex uses electromagnetic waves to measure the bulk apparent conductivity of the soil. Because it uses magnetic waves, it is susceptible to magnetic interference caused by objects that have a metallic nature or any current-carrying device because it produces a magnetic field of its own. Therefore, the environment around the sensor needs to limit these influences. This inspired the rule of thumb of a 1 m diameter, nonmetallic bubble around the sensor to guide initial testing of the data response. This bubble influenced the structural frame to be made of the carbon fiber crossbeams, connected by metal at the joints, so that the metal was as far away from the center of the robot as possible. The wheels are also placed in the corner and their metal is limited by making the hub from 3D printed PLA plastic. All wiring to the motors is carried on the outside of the robot, staying as far away from the center where the sensor is carried. Lastly, the electrical box is located at the back, along a crossbeam, away from the sensor. The result is a centrally located sensor, carried by carbon fiber and plastic crossbeams with metal no closer than 0.5 m away, or 1 m diameter.

2.4 Fabrication of Initial Prototype

Work building the first prototype began in year 2 of the 4-year EPSCoR wheat research project. The 3D printed PLA fender parts were printed on a Fusion 400 printer, each part taking a day to print. Once printed, the heated inserts were added to the structure, fastening the components together by Allen screws. The motor attaches to one side of the fender with the encoder on the other side, meeting in the middle for stability. The tread, made of a rubber-like TPU plastic, is fastened to the outside of the wheel with glue. An image of the designed fender and completed wheel fabrication is shown in Figures 2.5 and 2.6.

Figure 2.5

Fender Assembly

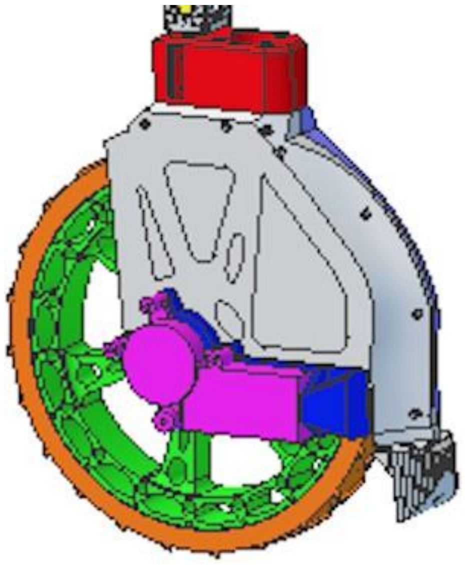
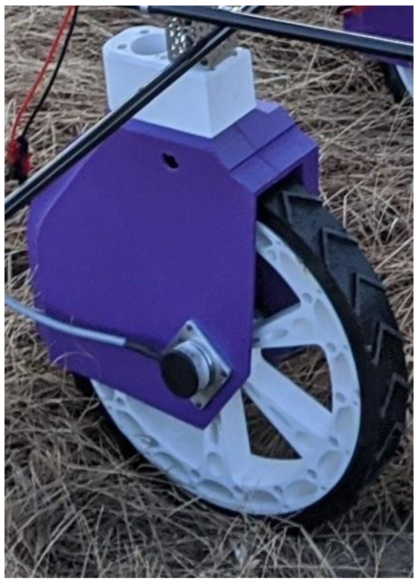


Figure 2.6

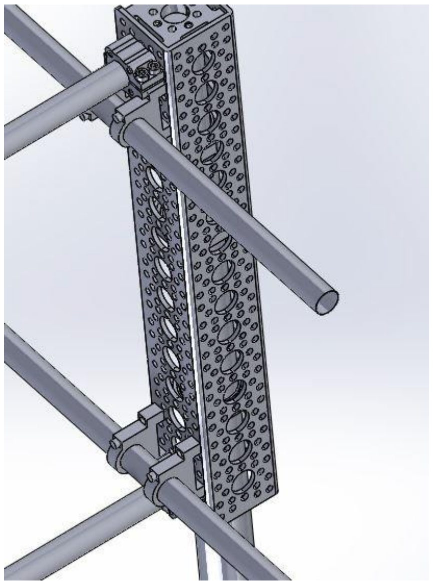
Fender Fabricated



After each of the four fenders were constructed, the carbon fiber tubing was attached to the aluminum C Channels via two aluminum bore clamps per corner for each tube. An image of the design and attachment is shown in Figure 2.7.

Figure 2.7

Robot CAD Model



The initial proposal used half inch carbon fiber tubing for the cross braces, as shown in the CAD model. However, it was determined that this would be insufficient to carry the robot's weight. The smaller diameter tubing was strong enough but would deflect with weight beyond an acceptable range. This was due to the small diameter contributing to a small moment of inertia. Therefore, it was proposed that by using this governing equation:

$$Rigidity = EI = E * pi * (d_o^4 - d_i^4)/64 \tag{2.1}$$

where E is the Young's Modulus for the specific material, I is the moment of inertia for the geometry, d_o is the outside diameter of the tube, and d_i is the inside diameter.

$$RigidityLargeTube/RigiditySmallTube = (d_{o1}^4 - d_{i1}^4)/(d_{o2}^4 - d_{i2}^4) \tag{2.2}$$

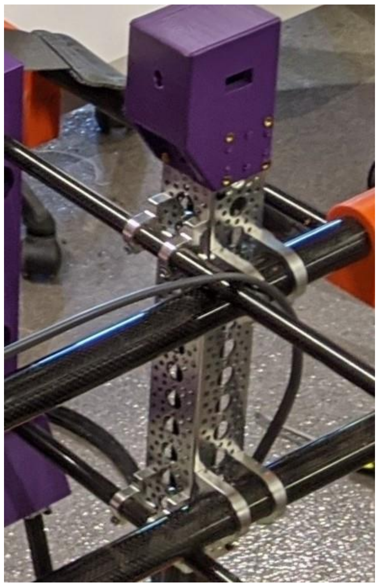
$$= (1^4 - .915^4)/(.5^4 - .435^4) = 11.203 \tag{2.3}$$

By expressing the change as a ratio between the large and small carbon fiber tubing, the Young's Modulus and the units both cancel out. The result of the ratio describes how much more rigid the larger tube is relative to the rigidity of the smaller tube. It is necessary to describe in this way as the Young's Modulus for carbon fiber can be difficult to determine because it is based upon the specific type of weaves used, the number of weaves and their orientation. However, by describing it as a ratio, it can be determined that the change in design will result in an 11.2 times greater rigidity of the frame.

The increase in diameter from 0.5 in to 1 in would lead to an increase in rigidity by more than an order of magnitude. The design was changed to incorporate this new concept, and the process of adjusting the design began (see Figure 2.8).

Figure 2.8

Robot CAD Model Initial Build



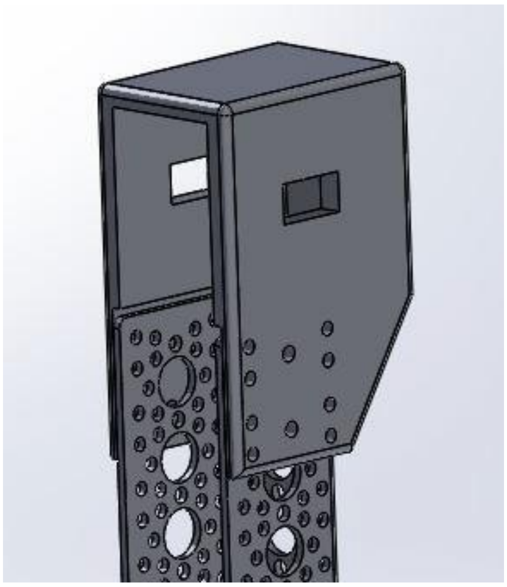
Note. Robot CAD model – one of the side lengths has been adjusted to account for the diameter change.

2.5 Prototype Steering Redesign

The initial design incorporated the use of motors located at the top corners of each C channel that would be used to rotate the shaft that extends down to the wheels. However, to use this design effectively, the motors would each need an encoder to precisely control steering of the robot. This element adds complexity to the design, increases weight, and overextends the limitations of the microcontroller PWM input and outputs from the MyRio. It would also require sophisticated PWM control through PID to precisely rotate the wheels to a specified angle while still being based on relative positioning instead of absolute positioning. This means that every time the robot is restarted, the steering motors would have to be calibrated based on their location. In combination with the four motors and encoders for each wheel fender, an additional four motors and encoders at the shoulders for steering would be eight total motors and eight encoders. This would require an additional microcontroller and more electrical wiring, which increase the number of electrical signals being sent near the sensor, altogether adding a great deal of complexity to the robot. To solve these issues, another design element was proposed with the encouragement from advisor Dr. Flippo that would use high torque servos to accomplish the task of steering (see Figure 2.9).

Figure 2.9

Servo Mount CAD Model

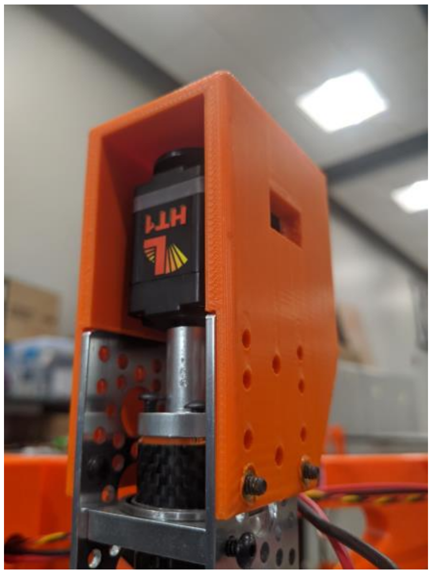


Note. This model uses high torque servos for steering to help with weight and other prior constraints.

The servos are small, lightweight, and easy to program. They can either be based on relative or absolute positioning. The drawback of the servo redesign is that the strongest servo found was still less capable in terms of power than the motors selected. However, the servos were determined to be powerful enough because as the robot propels forwards, the servos use the forward momentum of the robot to correct the wheel alignment with less force required. The change in design required a new form of attachment for the servos. The housing was designed by Calvin Dahms and is mounted on top of the aluminum C channel as shown in Figure 2.10.

Figure 2.10

Fabricated Servo Mount



The fabrication period was abruptly interrupted by the COVID-19 global pandemic during the winter of the first year. After a short hiatus from activity in the lab on Kansas State's campus, it was determined that the equipment needed to construct the robot could be put into a long trailer and moved offsite. The Fusion 3D printer and all necessary tools and components were stored inside the trailer (see Figures 2.11 and 2.12).

Figure 2.11

Remote Fabrication Studio

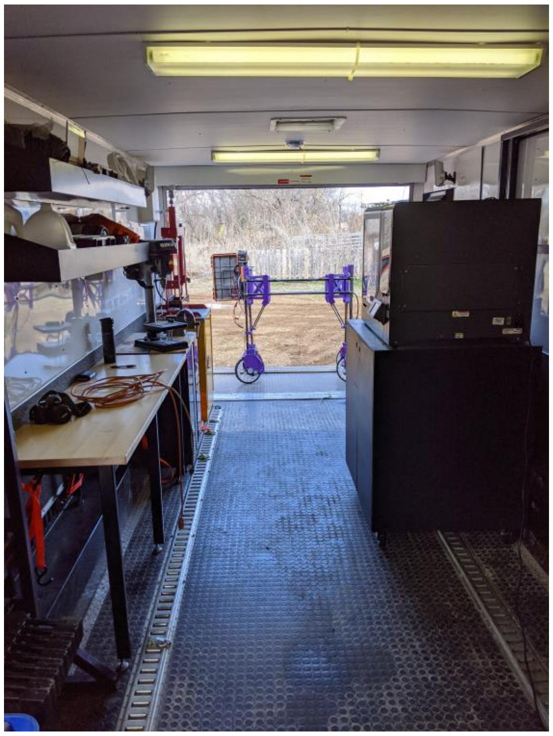
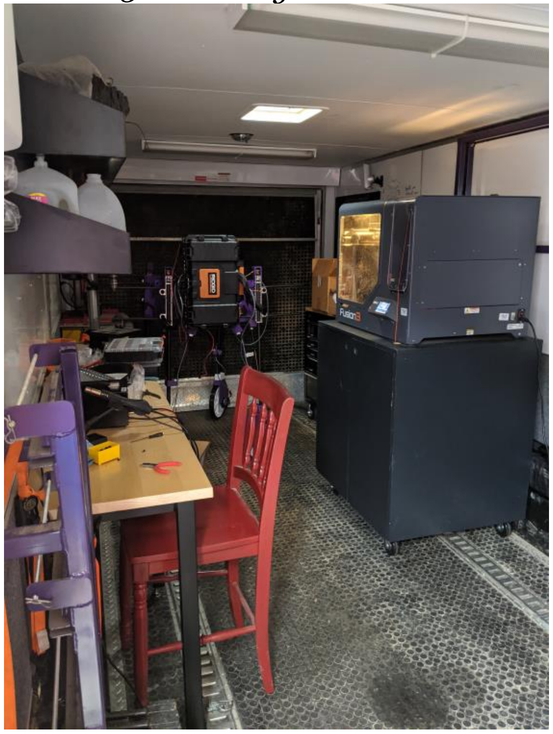


Figure 2.12

Remote 3D Printing

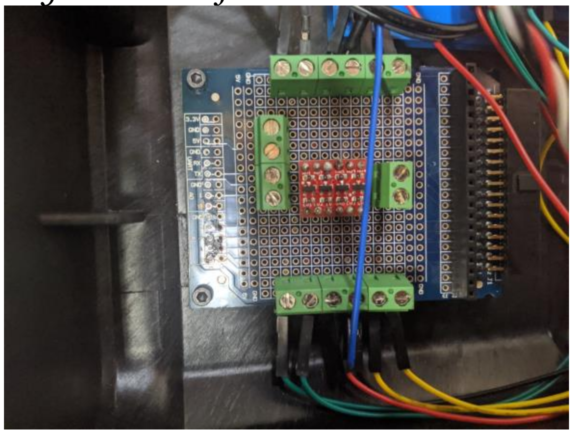


2.6 Prototype Electrical Box

To control the electrical components such as the motors, encoder, and the servos, an electrical box was designed to house the microcontroller, motor-controllers, voltage level-shifters, batteries, and voltage regulator. The design of the electrical box has the batteries feed into the voltage regulator that maintains a constant input voltage to the robot of 12 V. This incoming current gets split three ways: to the microcontroller, to the servos and to the motor-controller (see Figure 2.13).

Figure 2.13

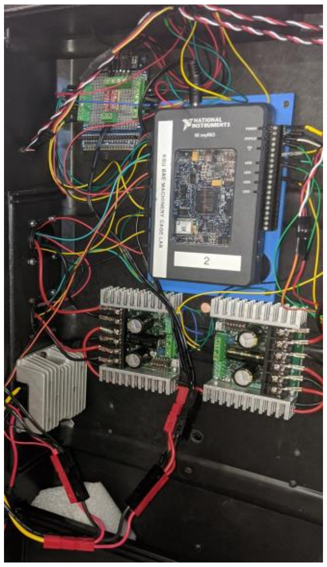
Logic Level Shifter



The PWM signal from the microcontroller is at 3.3 V and goes through the 8 channel bidirectional logic level-shifters (CYT1082) to maintain the 5 V logical PWM signal required by the servos for each individual servo. The PWM for the motor-controller is directly sent using 3.3 V and both are grounded between the 5 V constant supply from the microcontroller and the ground. A schematic of the electrical box is shown in Figure 2.27 ; Figure 2.14 shows an initial design.

Figure 2.14

Initial Electrical Box



The servos are put into the 360-degree rotational setting, which can be adjusted via the button-layout configuration for the LSS HT smart servo. At the same time, the Sabertooth 32 A motor-controllers are set to be controlled via PWM directly from a battery powered microcontroller as specified by Sabertooth.

Once completed, the electrical box was then mounted on the rear side of the robot, the furthest away from the sensor as possible, to limit electrical interference.

2.7 Prototype Testing

Once fabricated, initial tests were done by turning and moving the robot in the lab. These tests were to make sure the LabVIEW program was behaving how we would expect the controls to function.

The motor controllers are connected to the MyRio via Pulse width modulation that is controlled between 1 ms and 2 ms, with 1.5 ms being off and the two polls being positive and negative depending on the orientation of the incoming voltage and ground. The motor's PID motor control is constructed using LabVIEW's block style vi programming. The PID is constructed using the outgoing PWM control signal and the incoming encoder sensor data. The PID balances the two signals using control variables (Kp, Ki, and Kd) that determine the time and accuracy response of the control. The PID code was tested by lifting the entire robot onto plastic boxes, suspending the wheels in the air. The speed is set to a constant and the wheels begin to spin in unison. An object or a hand then restrict the wheels spin by adding a load by use of friction and the response of the PID is displayed on the computer.

Figure 2.15

PID Example Flowchart

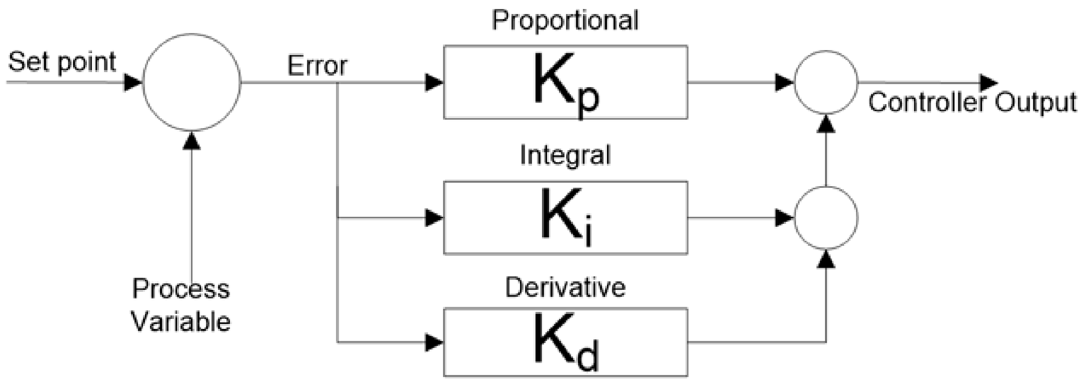
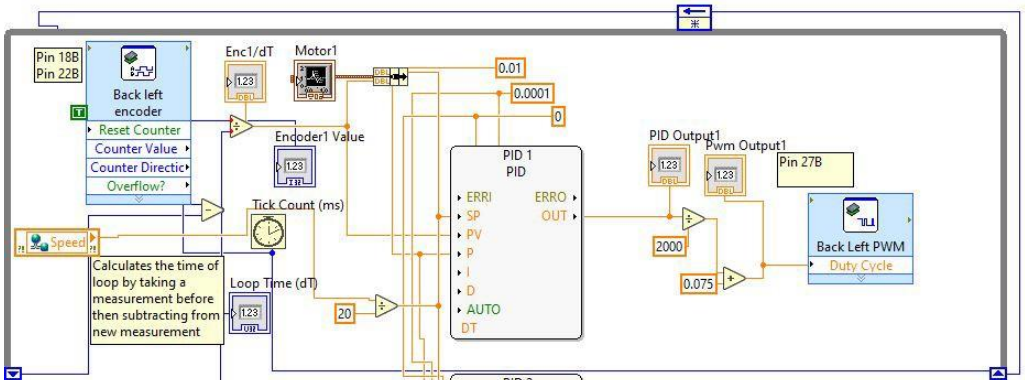


Figure 2.15 shows a simplified PID control system that is the backbone of the EPSCoR robot's motor control. The set point of the PID is the speed (not the voltage) that the robot should move at. The process variable is the PWM signal that is changed to determine this speed, and the error is calculated by reading the desired speed and subtracting the current speed from this. The current speed is calculated by the turn of the wheels being sensed by the encoders and measured as rotations per clock speed of the MyRio. This error gets fed into the three control constants of Kp, Ki, and Kd that determine new output. Kd is the derivative that responds to how quickly the error is increasing or decreasing, Ki determines the response based on how far away the speed is from the set speed and the Kp constant determines the response based on the proportional distance of the error to the set point.

The response time of the system is determined by a combination of this system, the predetermined time interval of the MyRio code, the limitations of the motors, and the resolution of the encoders. For the purpose of the EPSCoR robot, the precise nature of PID control that is possible is unnecessary. A general solution that can maintain a relatively constant speed is more than sufficient for the context of driving the motors in a field. If the motors do not abruptly alter the speed so that the direction of the robot is changed faster than the servos and GPS can respond and does not further inhibit the drivability, the PID control can be deemed sufficient. See Figure 2.16 for information on the code used.

Figure 2.16

LabVIEW PID Code



The main challenge of the PID design is determining at what speed each wheel needs to move. As the robot turns, the wheels will move at slightly different speeds. At smaller angles this usually can go unaccounted for and the changes to the robot's drive will be minimally altered. But because of the four-wheel drive nature of the EPSCoR robot, if the robot makes a pinpoint turn in the field, all the wheels will move at different speeds. The front wheels will move the fastest as they have the larger turn radius. Therefore, the code needs to determine at which speed each motor needs to be set.

2.7.1 Servo Control

To control the steering with the servos, constant PWM was used within the range of the servos to control the angle from 0-365 degrees. The result of this design change allows for the programming for directional control to be simplified to a PWM representing the direct angle of each individual wheel. Simply by changing the duty cycle of the pulses being sent to each servo, the absolute position of the wheel degree can be set. No feedback response or other sensors are required. Being able to control the direction of the wheels using just servos eliminates complex suspension, additional sensors, and other usual hardware used to turn. This simplification reduces the overall weight of the robot.

Once the drive controls have been sufficiently designed in the lab environment, the code is then saved as a real-time application stored on the MyRio microcontroller. This application boots up on the start-up of the robot. The robot is then connected to an iPad via Wi-Fi and controlled using a data dashboard.

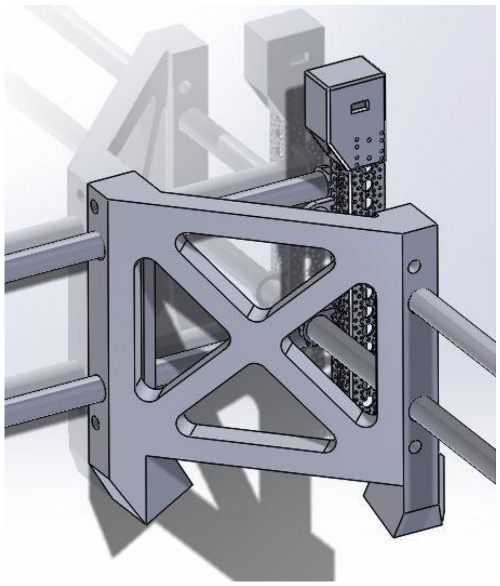
The robot was then driven outside behind the lab to run basic diagnostics of the wheels rotating under full load, the battery length, and the rigidity of the frame. The robot was driven around campus multiple times.

The results of these initial tests showed the gussets were needed for rigidity, the wheel fenders needed reinforcement, and the battery needed larger capacity.

2.8 Structural Redesign

The initial testing phase of the prototype robot immediately indicated a strong need for structural rigidity improvements. The robot tended to bend and compress the corners opposite each other, creating instability and restricting driving capabilities. Shown in the previous testing images, gussets were designed at the corners to address this concern. Initially it was thought that putting a brace at the corner, horizontal to the ground, would address the corners compressing. This design element is shown in a CAD model (see Figure 2.17) and was tested. Although it helped alleviate a portion of the problem, it did not solve the issue entirely.

Figure 2.17
Gusset CAD Model

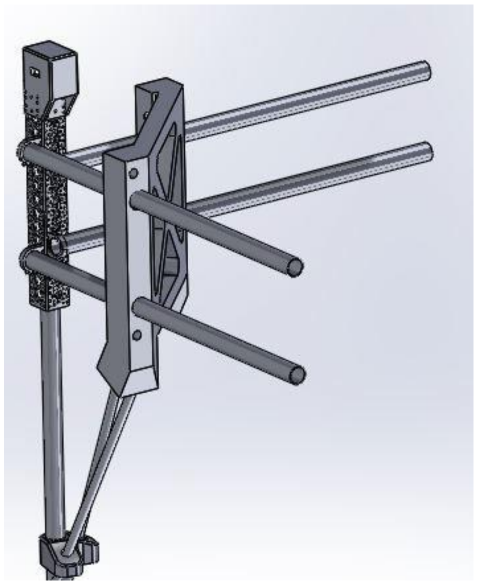


The result was an effective change in the bending arm from 1.67 m to 1 m from corner to corner. This helped diminish the overall deflection due to bending but did not eliminate it.

This was primarily because the bending of beams was not just happening with the crossbeams. After 3D printing and constructing these gussets, the issue was revealed to be more pronounced in the bending of the wheel shaft that extends down toward the fenders. In hindsight, this makes sense, as most of the resulting force on the beams is from friction in response to the movements of the robot through the field, whereas the horizontal crossbeams experience less friction and primarily static weight. This led to a redesign that incorporated the gusset but included the downward extending shafts toward the wheels in a triangular pattern (see Figure 2.18).

Figure 2.18

Gusset with Reinforcement CAD Model



The main benefit of this design was the immediate reduction in the bending of the moment arm of the wheel shaft. Originally, this moment arm was 1.33 m long. With the addition of the gusset braces that connect to the shaft as shown in Figure 2.18, the moment arm is reduced to 0.67 m. The result of the change was a reduction in deflection from the initial 7.5 cm to just 1.25 cm, a 6x reduction.

Figure 2.19

Gussets Being Mounted for First Time

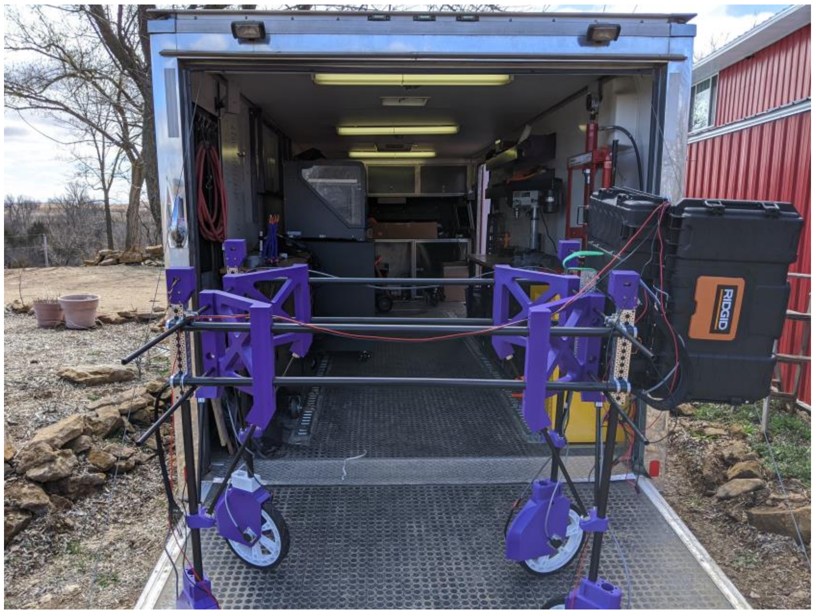
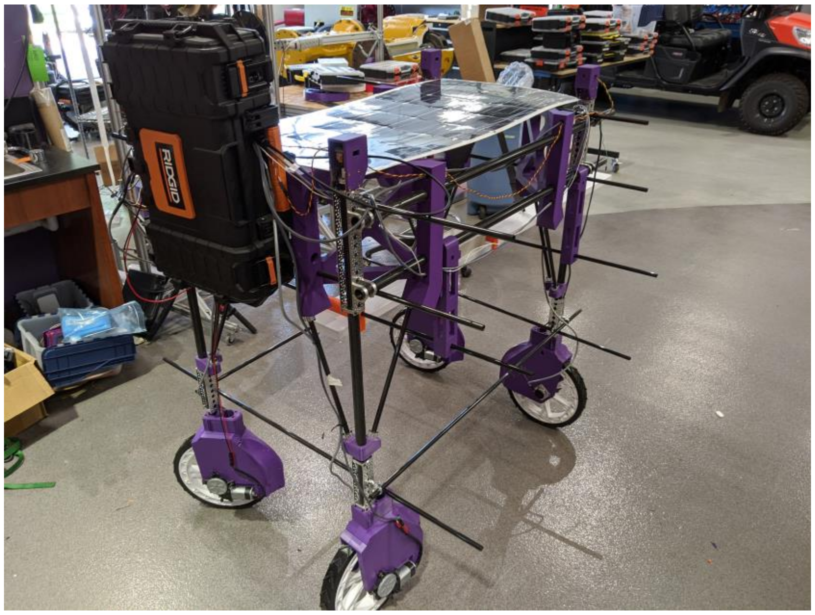


Figure 2.19 shows the designed gussets attached at the remote lab. Each corner of the robot has its own gusset that connects to the according wheel shaft. The result was greatly increased rigidity of the frame, as expected.

Additional design changes incorporated solar panels into the design of the robot. The panels were part of the initial proposal for the robot to be able to charge in the field and operate on a seasonal basis. As designed, the panels lie flat on top of the robot, attached by zip ties to the frame for flexibility (see Figure 2.20). The panels themselves are flexible plastic sheets with the solar cells attached to allow for the panel to bend appropriately. It was easy to adjust the position and attachment of this type of panel relative to the robot itself, which is important as the design of the robot, its length and width, changed throughout the design process.

Figure 2.20

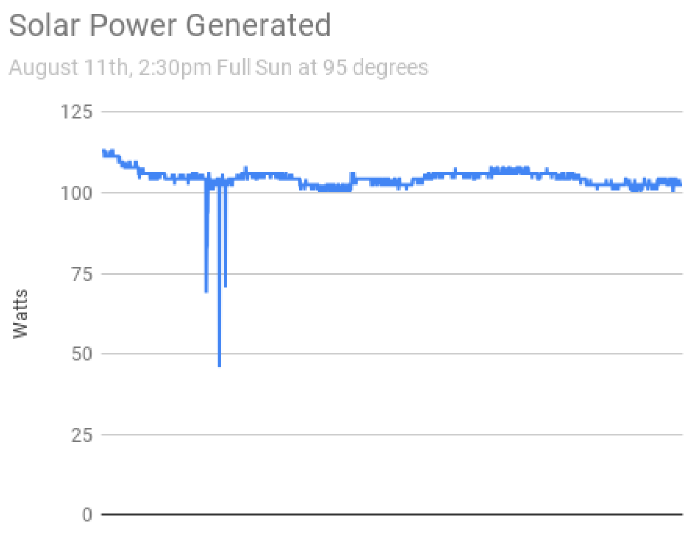
Mounted Solar Panels



The panels are 18 V 100 W solar panels operating up to 7 A of current in full sun. The test showed that a more reliable estimation of the power from the panels received by the robot would be a power capacity of just over 50% compared to the specified power of the panels. The test data is shown below that between two panels connected in series, the power received was 104 W compared to the rated power of 200 W of the panel's combined power. This is relatively normal for solar panels as the rated power is in perfect lab conditions whereas the solar panels on the robot are flat, horizontal to the ground, not perpendicular to the sun, and even full sun conditions have less power than lab conditions for testing. Figure 2.21 charts the amount of power generated under full-sun conditions.

Figure 2.21

Solar Power Generated Under Full Sun

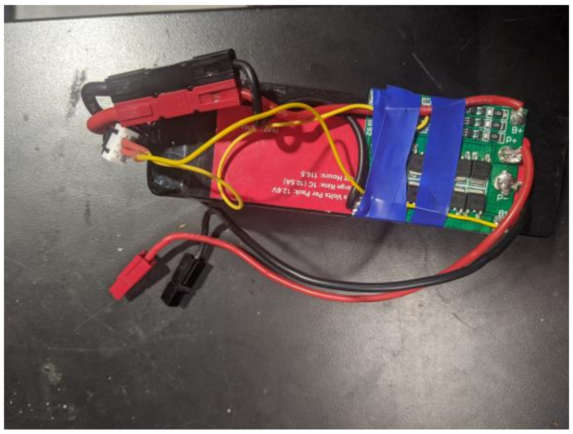


The solar panels require the use of a solar charge controller to manage the incoming power from the panels and distribute it to the robot. The controller handles voltage from solar panels of up to 36 V with an output of 24 V to the robot, which fits perfectly into the design of the robot. The robot incorporates two 3S venom 13500 batteries in series, amounting to 24 V. And the two get sent through the voltage regulator, which requires 24 V input from the batteries and solar to maintain a constant down shift to 12 V to be distributed through the robot.

To protect the batteries from being overcharged from the solar or over discharged from the robot, the 3S battery protection board (BPB) is attached to each battery and monitors each cell and balances the incoming power to protect and maintain battery health (see Figure 2.22). If the battery charges above a certain voltage, the BPB shuts off charging; if it drops below a particular voltage, it stops discharging.

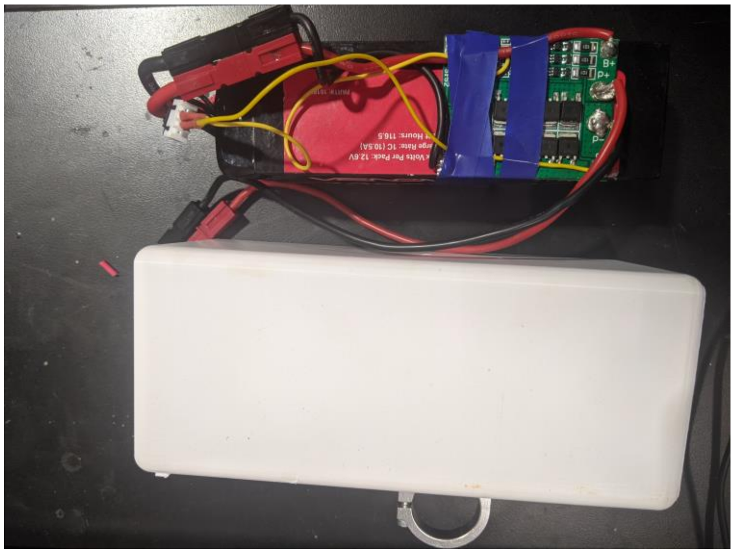
Figure 2.22

Battery with Battery Protection Board



The weight of the batteries is then distributed across the robot using a battery compartment that can be attached to any of the beams (see Figure 2.23). This limits the weight from being located in the electrical box, reduces strain on components such as the servos, and increases performance of the robot in terms of power and steering capability.

Figure 2.23
Battery Compartment

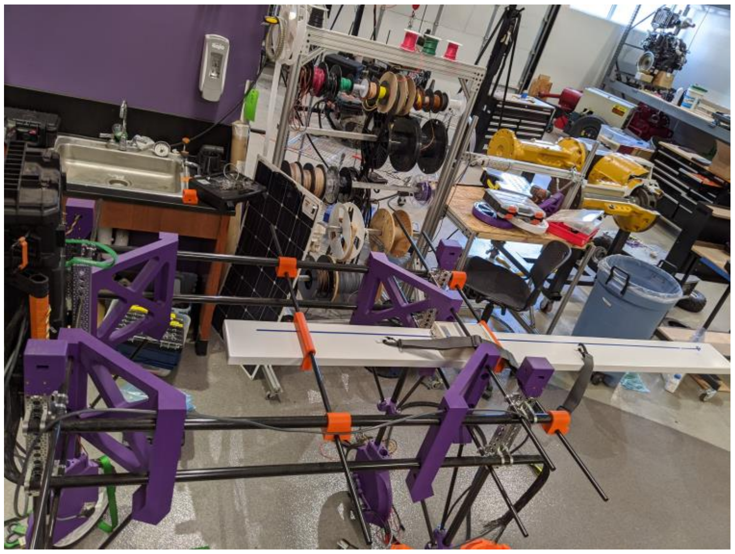


The compartment is hollow and allows the batteries to slide in and out to charge. It also is self-locking using the weight of the robot, which protects the battery from being exposed to the elements.

Another new design added to the prototype was the EM sensor mounting system. Initially, this mounting was at the top of the crossbeams, just under where the solar panels are now mounted. They were designed to use a 3D printed plastic part that would connect a carbon fiber crossbeam that spans the middle of the robot, threading through a mounting brace on the bot to the front and back of the sensor (see Figure 2.24).

Figure 2.24

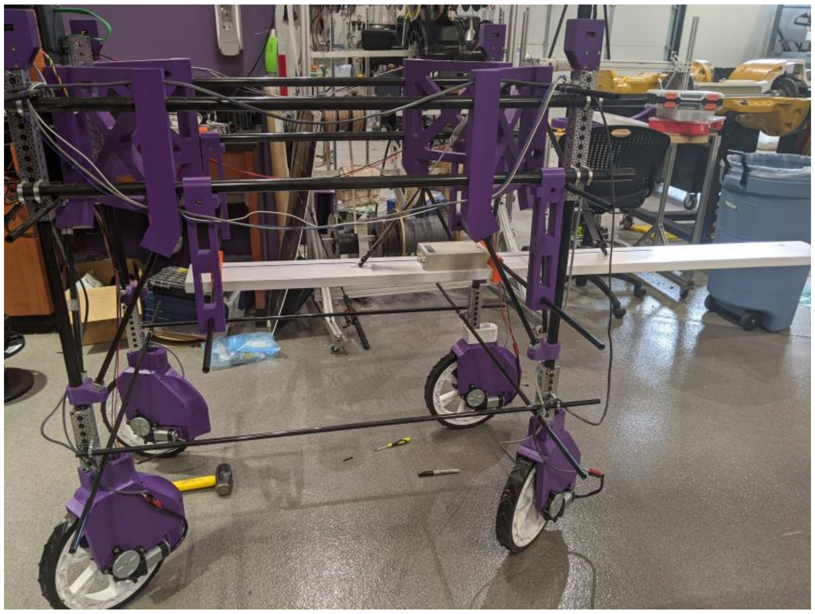
Initial Sensor Mount



However, with the addition of the solar panels, this design was no longer feasible as it would place the sensor too close to the metal of the solar panels. To alleviate this problem, the sensor mounting system was redesigned. The sensor was placed on the lower carbon fiber crossbeam and the conversion was extended downward by an additional 33 cm (see Figure 2.25). This change maintained the previous constraint of the half meter bubble where metal could not be located.

Figure 2.25

Sensor Mount Redesign



Additionally, as seen in Figure 2.25, there were carbon fiber crossbeams that were originally designed to be just above the fenders. With the addition of the gussets and the braces connecting to the wheel shaft, these lower cross braces would later be removed, reducing weight.

2.9 Updated Electrical Box

The last major update to the robot platform at this stage was the addition of the LattePanda Alpha computer. This computer is a combination of a Windows processor and operating system with a built-in Arduino. Jonah Smith originally chose this because the EM sensor from GeoPhex requires the Windows platform to be able to read and interpret data. And for the robot to respond, it needs serial exporting capabilities through the on-board Arduino. This computer was added to the electrical box and included in the communication protocol of the robot. Jonah was successfully able to read data from the sensor using both serial and Bluetooth connection by incorporating an out of the box MatLab program supplied by Geophex. Once the capabilities to read the EM sensor had been accomplished, Calvin Dahms wrote the program to transfer the data to the robot via serial link and indicate successful interpretation by an LED light indicator on the MyRio. This data was then to be saved on the hard drive as well as interpreted by the robot and saved to a flash drive. The LattePanda computer would later open the door to more robust forms of processing and control methods for the robot, and the communication methods would ultimately change. But at this point, the sensor had been demonstrated to record viable data and transmit the data to the robot's processor.

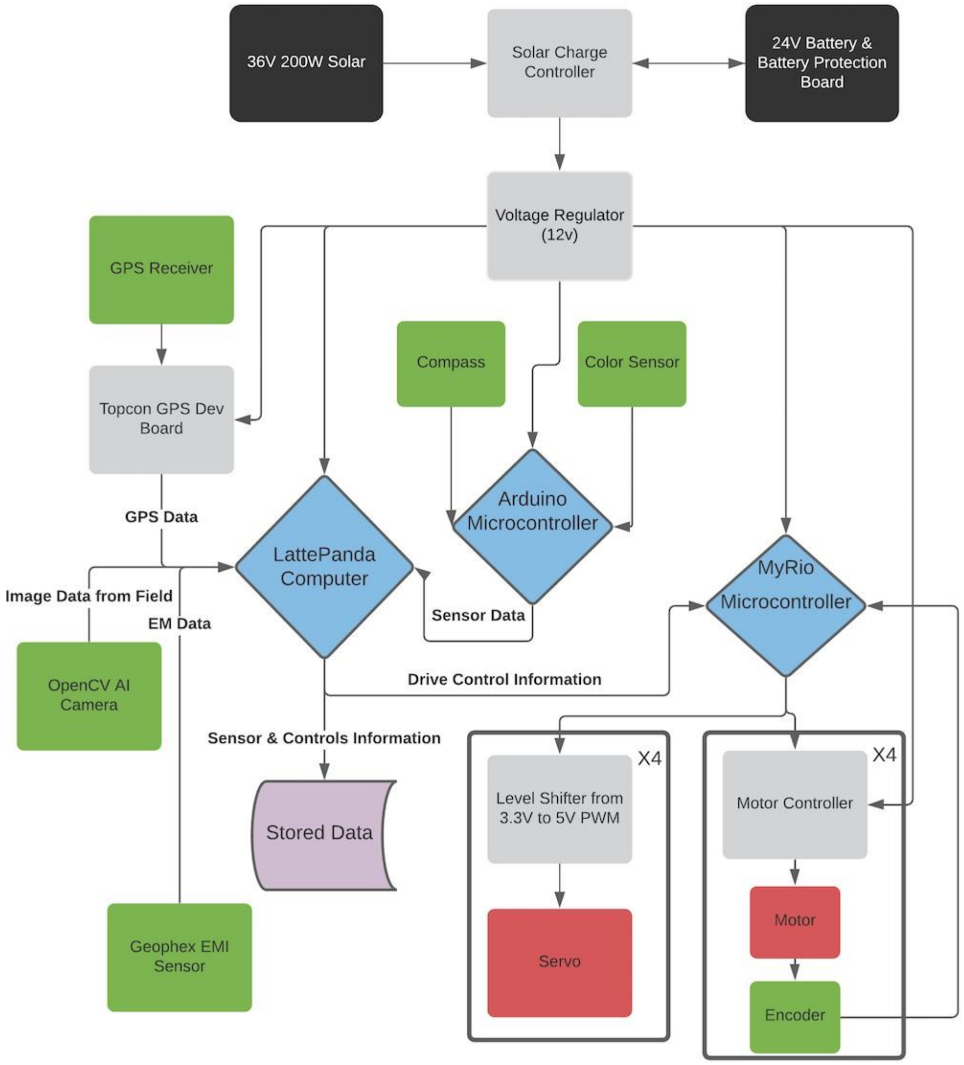
The inclusion of the new computer also aided in the accessibility to the Topcon B125 RTK GPS and Topcon Developer Board that connects to the computer via serial. The connection is relatively simple to the computer and the data can be robustly interpreted using libraries within the python language. The GPS data can therefore not just allow the sensor to pinpoint where each data-point is coming from but can also be used to calculate waypoint and heading of the robot. To receive the signal of the GPS and be able to interpret and compare the data, an antenna with the GPS receiver and a corresponding magnetic field compass sensor are attached above the electrical box.

The main compartment of the electrical box has the MyRio arranged at the top of the box alongside the power and signal distribution for the servos. Just below the MyRio are the two large motor controllers. Mounted on the left wall is the golf cart voltage regulator that steps down the 24 V from the battery and solar charge controller to the usable 12 V for the robot.

Mounted on the door are the LattePanda Computer and Arduino, the Topcon GPS Developer Board, and the solar charge controller. Each of these directly connect to each other and are therefore mounted in relative proximity. The batteries are located on the opposite side of the robot in their own 3D printed housing to balance the weight of the robot. Figure 2.26 is a schematic representation of the electrical box components.

Figure 2.26

Robot Electronics Schematic



- Power in Black
- Controllers in Grey
- Sensors in Green
- Drive Train in Red
- Computers in Blue

Figure 2.26 displays the flow of power originating from the solar panels, with excess power stored for later use in the batteries. The direction of the power is controlled by the solar charge controller and fed through the 12 V regulator to bring the voltage down to the required 12 V power for the robot components. The LattePanda computer, Arduino, MyRio, Topcon GPS Developer Board, motors, and servos all run on 12 V architecture to simplify electronics and maintain adequate power for driving.

The computer controls all the high-level logic, reading information from the sensor suite and determining the method of control to the MyRio. The Arduino reads the data from the compass and color sensors and passes them onto the computer through the serial bus. The computer then compares the values to the positioning data from the GPS. The absolute positioning from the GPS is used alongside the incoming data from the relative data from the images, and environment sensors to determine the speed and direction of the robot and stored as data on the hard drive.

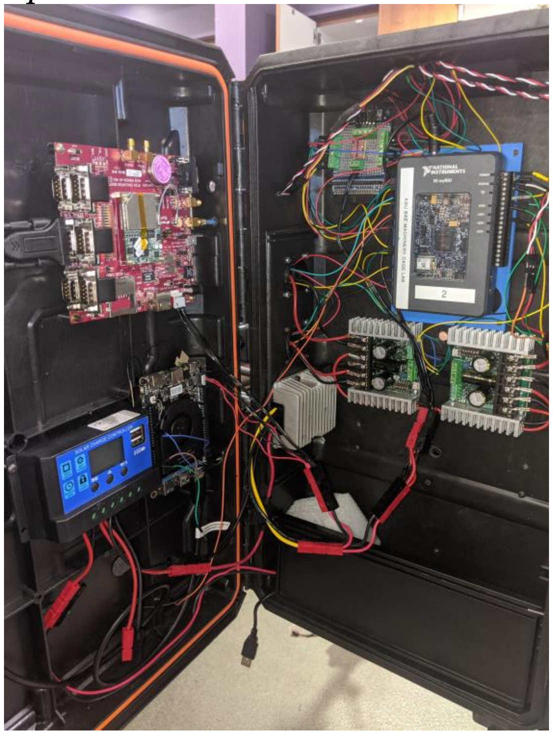
The information is then passed to the MyRio microcontroller, which interprets the information and converts it to PWM commands. These commands are passed to the servos and motor controllers to move the robot through its environment. Figures 2.27 and 2.30 show the updated electrical box with the LattePanda, the GPS, and the Solar charge controller on the left side.

Figure 2.27

GPS and Compass Antennae

Figure 2.28

Updated Electrical Box



2.10 Wavelength Detection (Color) Sensor

In addition to the EM sensor, the robot also incorporates a color-detecting sensor designed by Langston University student Jordan McClellan. This sensor's intended purpose is to read incoming light waves to determine what color object is in front of the robot. The incoming data from the light wave meter would be read by the robot to avoid running over obstacles the color of wheat, such as green and yellow. The sensor would be mounted on the wheel fender facing forward on the robot. A picture of the sensor is shown in Figures 2.29 and 2.30.

Figure 2.29

IR Sensor

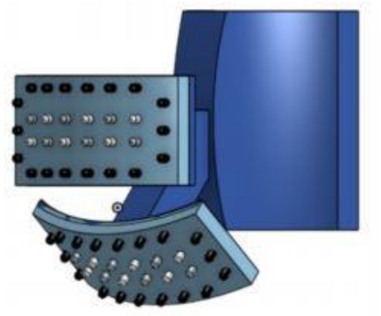
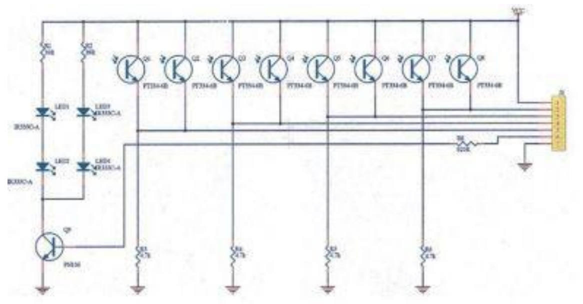


Figure 2.30

IR Sensor Schematic



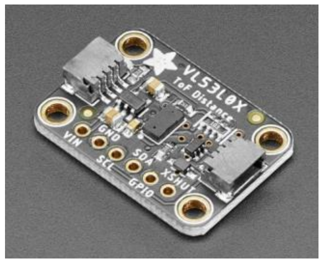
The sensor proved feasible in a lab and could detect a change in wavelength from brown to green to gold. However, some challenges persisted with the sensor when lighting conditions, such as a cloud, change the amount of incoming light. This change in light reflected would give a faulty reading that differed for the same color. For example, under less light, yellow may be perceived as brown. This was due to the nature of the data being represented as a relative magnitude. In addition, the sensor in its current state required many wires with complicated coding to read and interpret the incoming data. Each emitter and receiver has its own data line and would need to be compared against each other to produce meaningful data. To solve these two challenges, it was proposed that by using a TRIAD SPECTROSCOPY SENSOR AS7265X by Sparkfun, the amount of light coming in at a particular moment could be used to calibrate and respond to changing lighting conditions. The sensor's placement and housing were designed in collaboration with Calvin Dahms and Ben Weinhold to place the sensor above the wheel pointed downward at an angle to read 1 m in front of the wheel. Alongside the light reading sensor is placed two Adafruit VL53L0X Time of Flight Distance Sensors (see Figures 2.31 and 2.32).

Figure 2.31

TRIAD SPECTROSCOPY SENSORAS7265X

Figure 2.32

Adafruit VL53LOX Distance Sensor



These sensors use a laser range finder to detect distances of up to 1200 cm. The resulting data records the color, the direction, and the distance of an object detected in front of the robot. The robot then takes this data and responds by stopping or turning away from the detected object. See Figures 2.33 and 2.34 for more tests and results.

Figure 2.33

Campus Tests

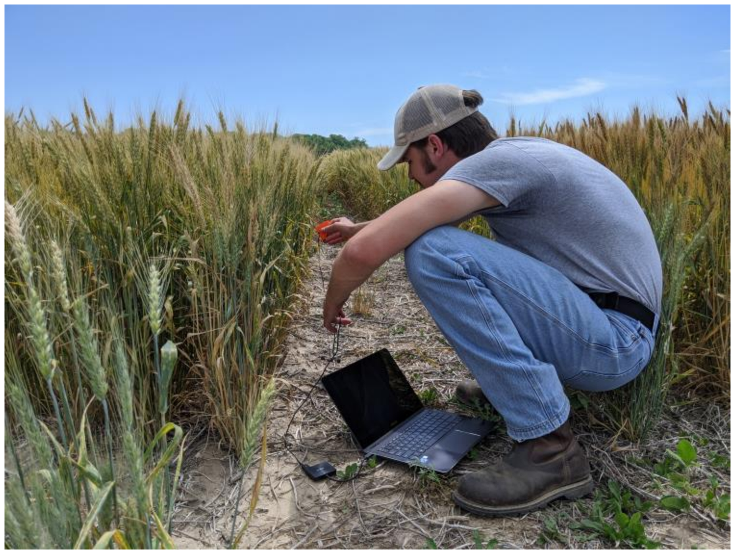
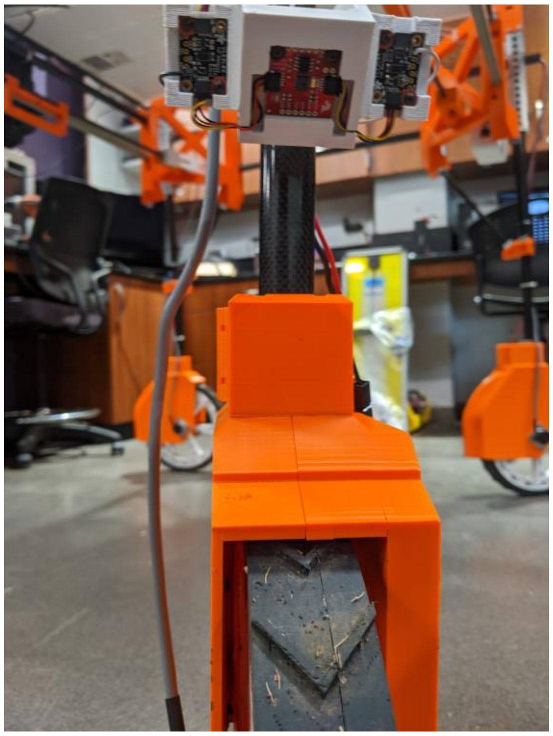


Figure 2.34

Visible Spectrum Sensor with Auxiliary Distance Detectors

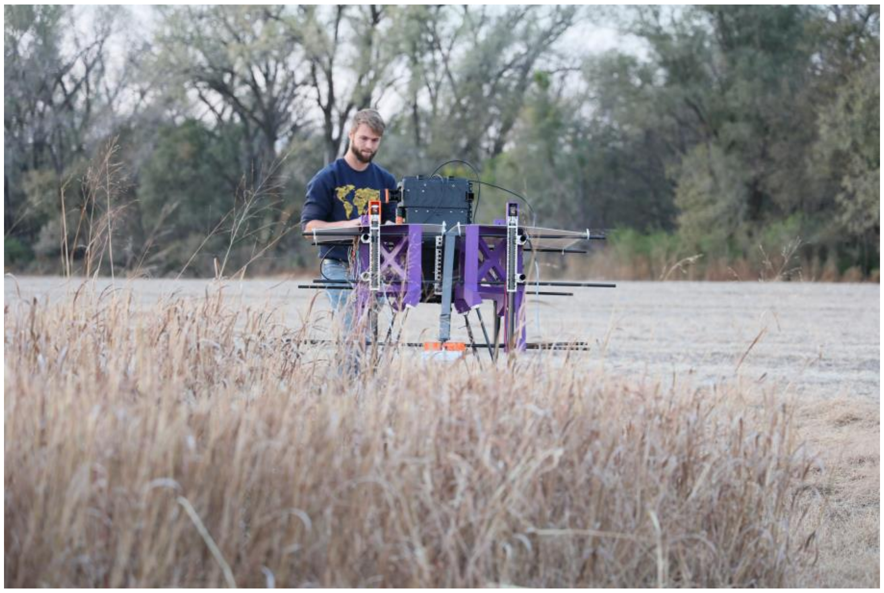


2.11 Testing the Redesigns

The redesigned prototype underwent testing at the North Agronomy fields, just north of Kansas State University's campus, and the Ashland Bottoms research fields (see Figure 2.35). The updated rigidity of the frame and the sensor mount were primary concerns, and the new design performed well in the environmental conditions. The gussets with the added braces increased the overall rigidity of the frame, improving the driving performance by limiting the compression of the corners to a minimum. The designed mount for the EM sensor performed well and was able to carry the sensor with slight vibrations through the field that are manageable.

Figure 2.35

Initial Field Testing

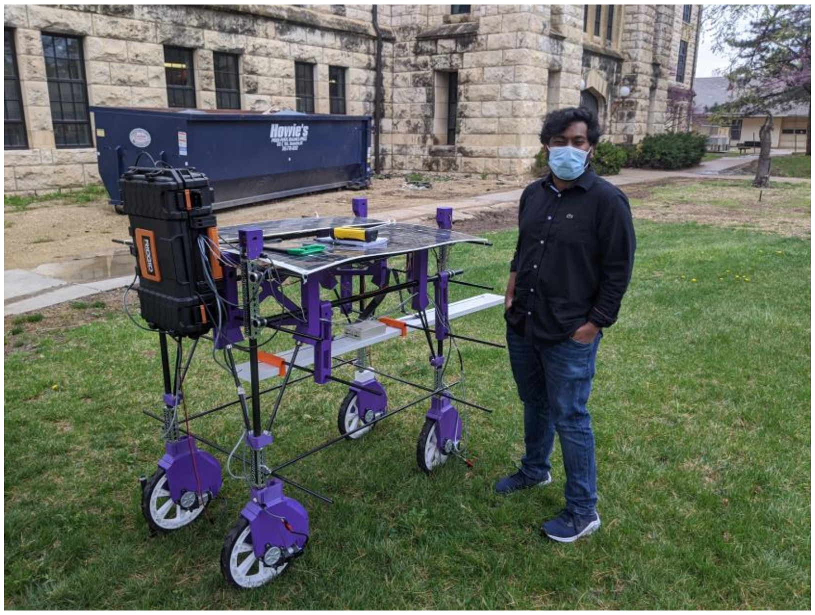


Note. Field tests in the fall of 2020 were done at the Ashland Bottoms and North Agronomy research fields to determine the functionality and drivability of the robot.

The tests performed in fall of 2020 were entirely about the functionality and drivability of the robot. The robot could traverse the rugged terrain within the target environment and drive through high foliage. The tire tread overcame inconsistencies with traction, and the power from all four wheel was shown to be entirely necessary when a wheel lifted off the rugged ground.

Figure 2.36

On Campus Drive Tests

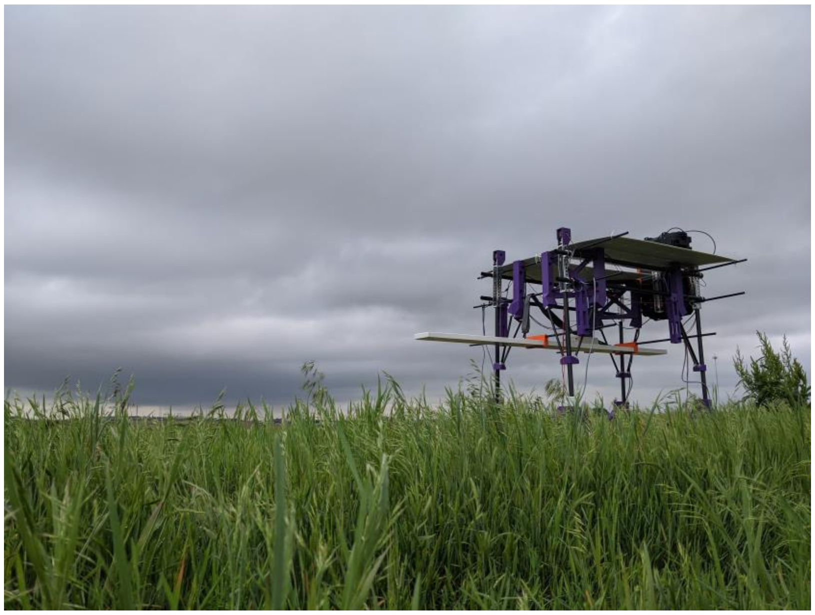


Note. Tests done on campus focused on testing the functionality of recording the data.

This image was taken while Calvin Dahms and Saquib Haroon performed tests on campus, behind the engineering lab. These tests were primarily to test the functionality of recording data from the sensor at specified distances and speeds for reliable, consistent data. A 50 m line was drawn, and the robot would carry the sensor as it drove along the line for multiple passes. Data was collected while the robot drove itself, while the robot was carried and turned off, and with a human carrying the sensor. This data was intended to show any alterations in the signal from the sensor that the metal from the robot might cause. However, it was determined that the data was not helpful because of unseen power lines in the ground that greatly affected the incoming data. The robot was then put into the truck to be hauled off campus for the remainder of the sensor tests to not have this interference moving forward.

Figure 2.37

Drive Tests at the North Fields

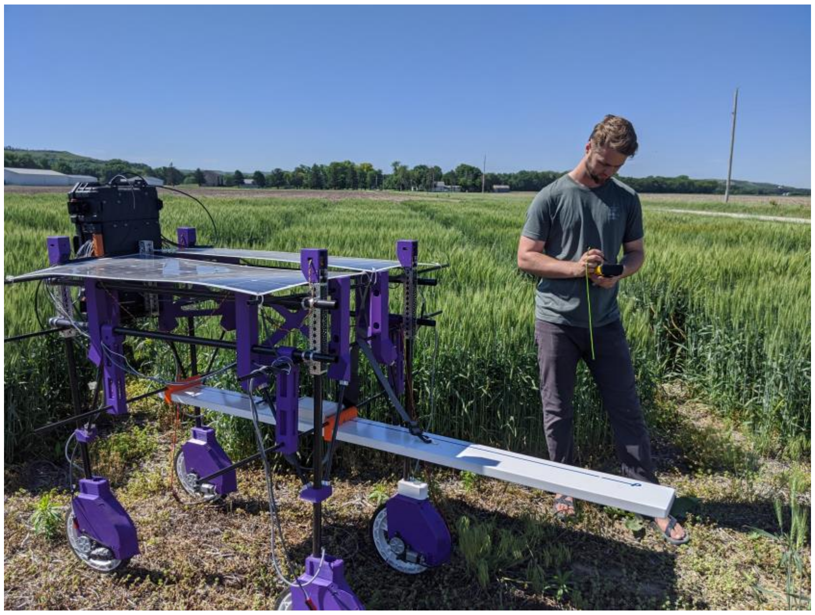


Note. Drive tests at the North Agronomy fields also sought to test data recording accuracy.

The image in Figure 2.37 was taken at the North Farm field. Researchers conducted the same data collection tests as they had on campus: driving the robot, carrying the robot, and hand-carrying the sensor. The data received showed a variation that seemed as though the sensor was recording reliable data that could then be used. The team then decided on methods for testing and calibrating the sensor at the Ashland Bottoms research field south of Kansas State.

Figure 2.38

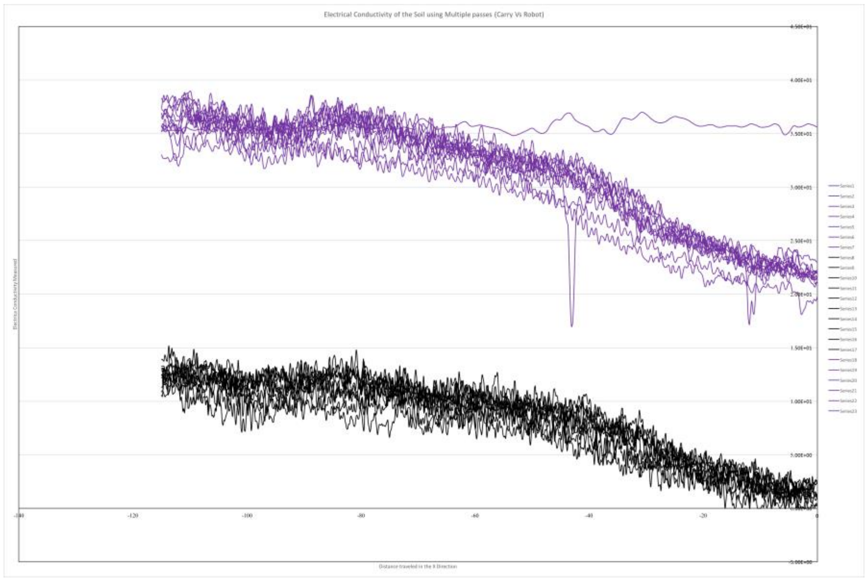
Drive Tests at Ashland Bottoms Wheat Field



Note. More extensive data testing was conducted at the Ashland Bottoms fields.

A similar test was conducted at the Ashland Bottoms field (see Figure 2.40), but with many more passes and over multiple days to see different inputs and their respective outputs. The image above shows the robot alongside the field of wheat where the data was taken. The robot would drive for 50 m for multiple passes and then the sensor would be carried along the same path, the same number of times. The resulting data is shown in Figure 2.39. It appears from the data that the robot influences the sensor by creating an offset in the data. The purple lines are passes the robot took carrying the sensor, and the black passes are where a human carried the sensor. It is clear from the data that the robot records consistently precise data that is not accurate. The thought process coming out of these tests is that an offset determined by a series of calibrations could determine if the measurements are predictable and useful. However, later tests would demonstrate that while there is an offset within the data, the offset fluctuates based on the conditions in the environment, such as temperature and moisture. Therefore, using a calibration method, while still possible, is much more complicated. It would also require initialization for each subsequent test to be able to interpret the data.

Figure 2.39
Initial Sensor Data



Note. More extensive sensor testing showed skewed results. The purple lines represent when the robot carried the sensor. The black lines represent when the researchers carried the sensor.

Additionally, during the wheat field tests, it was shown that while the robot could navigate through the field in its present state, the tolerances within the wheel path were too tight. The overall width of the robot allowed for less than 5 cm of spacing for each wheel to turn within a wheel path width of 30 cm. It would be better to have the wheel be centered within the wheel path, requiring a width increase of 50 cm. Otherwise, the tests showed the capabilities, especially structural, were proficient in the field.

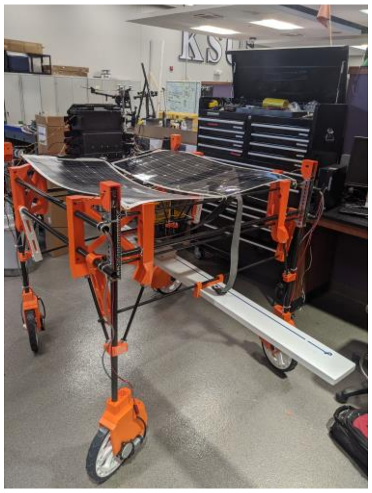
2.12 Fabrication of the Second Robot

The project is a partnership between Kansas State University and Oklahoma State University, so therefore, a second robot was fabricated, replicating the same technology as the first. This second robot would use the orange color scheme instead of the purple of the Kansas State robot in the 3D printed parts, otherwise the bill of materials remained almost entirely the same.

The only major adjustment to the robot was the exchange of the carbon fiber crossbeams for a longer, 1.67 m fiberglass crossbeam. This alteration allows the wheels to be situated directly in the center of the wheel path. A reason for the change in material came from the testing and consideration that carbon may impact the sensor, it was also much cheaper and easier to acquire on shorter notice. The analysis of carbon was as a reflection of the initial proposal's use of carbon fiber. The bubble of influence of the sensor was designed to have no metallic materials within a half meter of the sensor, however, carbon can act similarly to metal in terms of electrical interference because of its conductive nature. Therefore, it was better to incorporate a fiberglass tubing structure of the same diameter because it would perform electrically better and be cheaper and easier to acquire. Because it was the same diameter, no other design changes were necessary to make the switch.

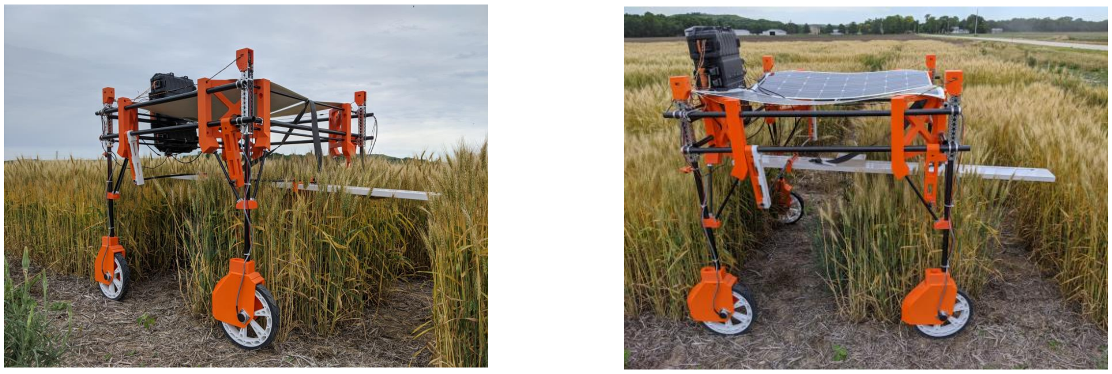
Additional changes were minor and only regarding auxiliary sensing capabilities. During the fall of 2020, Calvin Dahms determined it was feasible to design an algorithm to detect and label in-field objects while the robot was in the field. With the addition of the LattePanda Alpha computer, the opportunities for more advanced programming were possible and explored. In the new year, the algorithm was completed with the help of Sujith Gunturu using a 1MP camera. Once the feasibility of the wheat detection algorithm was proven, Sujith took further steps to improve the efficiency and capability of the algorithm and its uses, while Calvin designed the mounting mechanism and the methodology for how such an algorithm could be practically used. Further discussion on the camera and object detection is discussed in the section "Wavelength Detection and Image Processing." Once fabricated, the new Orange Oklahoma State robot was taken to the field to test its capabilities in the environment (see Figures 2.40 and 2.41).

Figure 2.40
Orange Robot in the Lab



Note. Oklahoma State University created a second, nearly identical robot to the one created at Kansas State University. It is referred to as the Orange Oklahoma State robot.

Figure 2.41
Orange Robot at Ashland Bottoms



Note. The orange robot conducted similar tests to the purple one at Ashland Bottoms research fields.

2.12.1 Serial Communication

The controls of the robot through the MyRio are communicated through a Serial communication link from the MyRio to the LattePanda and the Arduino. Each receives commands that are specific to the needs of the controllers to drive the robot or sense its surroundings. The MyRio takes inputs from the LattePanda and the Arduino that indicate the desired outputs for robot speed and direction. In return communication, the robot sends all relevant logged data by the MyRio, data such as actual speed from the encoders, power usage of the robot, and battery storage levels.

This communication is then passed to and read by the other two controllers to be used in their calculations determining speed, precision of the response to the commands, and logged storage of the data. All data that is relevant is stored for later retrieval to better understand the robot and the robot's environment.

2.13 Arduino Sensing and Communication

The Arduino acts as an intermediary between the MyRio and the LattePanda while executing some analysis itself. The Arduino Receives commands from the LattePanda, passing them to the MyRio, and receives MyRio data, returning the message to the LattePanda to record and store. Additionally, the compass is connected to the Arduino for the Arduino to make heading calculations for the robot. Future sensors can be added to the Arduino's pin commands if it is simpler or more pin connects are desired than what the MyRio can handle.

2.13.1 Compass Reading and Analysis

The Pmod2 compass sensor records the magnetic influence of its surroundings in three dimensions. It takes this data and using Pmod's libraries it calculates the true heading of the robot. The Arduino then takes this heading and compares it to the desired heading to the next waypoint. The result from the calculation is the difference between the heading and the desired heading and output as an angle. This angle is the angle data that is sent to the MyRio to control the wheel direction.

2.13.2 Power Sensing and Logging

Additionally, the Arduino measures the voltage of the battery and the current coming from the battery and from the solar panels. The voltage of the panels is held constant because of the nature of solar acting as a current supply. The Arduino connects to these sensors using its pin out and sends the received data to the LattePanda via serial link along with the compass data and the data received from the MyRio, such as the robot's speed and wheel angles.

2.14 LattePanda Advanced Programming

The LattePanda requires more sophisticated coding through Python, but in return, it allows for more robust sensing and analysis, as well as data logging into its internal memory. In addition, a USB memory card can be used to store and extract the data for laboratory analysis. The LattePanda received the end command of the data collected from the MyRio that passes through the Arduino and attaches the Arduino's communication. The result is that position, speed, heading, battery information, and any other relevant data is all processed and stored in one location on the computer. See Figure 2.42 for an example of the LattePanda.

Figure 2.42

LattePanda Alpha

The LattePanda handles all the path planning of the robot. This is because the GPS is directly connected to the computer and the planned route is derived from planning the waypoints. In addition to the RTK GPS and waypoint path planning, the LattePanda also uses an object-detection program through Python TensorFlow.

2.14.1 RTK Reading

Figure 2.43
Topcon Developer Board

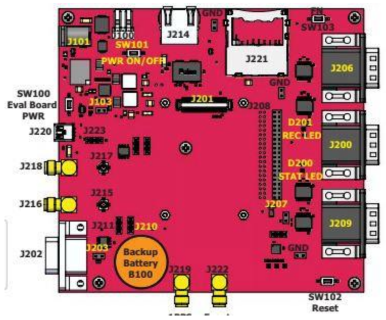


Figure 2.44
GPS Receiver

The Topcon B125 RTK GPS is read via Rs232 to Serial protocol from the B port of the Topcon Developer board. The python script accesses the serial port and reads the data that is constantly streamed from the GPS board. It flushes the old GPS coordinates and records the new position. This position is then compared to the desired positional way-point coordinate. Distance and heading are then calculated in respect to the location of the robot in reference to the desired place in space. An example is shown in Figures 2.43 and 2.44.

Once the distance and heading are calculated, the robot then determines if it is within a specified buffer of the waypoint. If it is within the buffer, then the robot selects the next waypoint in the predetermined GPS path by the mission planner. The new direction and heading are then calculated and sent to the robot.

2.14.2 Wavelength Detection and Image Processing

The object detection protocol was developed because of the high value of the research wheat and strict constraint to never run over the wheat. To accomplish this feat, the robot must keep its wheels within the designated paths between the research plots. If the robot is off by more than 10 cm, the risk is great that wheat will be run over. The RTK GPS is accurate within 1 cm, however redundancies are needed in case the waypoint tracking of the robot is not consistent enough.

In addition, the goal of the robot is to not just record data in the field but to do so consistently and to over time reduce the need for human involvement in the field. To accomplish this feat, the robot must become more and more aware of its surroundings through sensors.

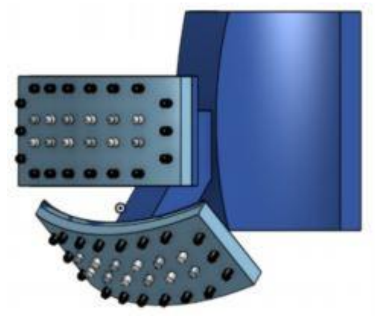
The first, most fundamental sensors to help with in-field path planning are the encoders, the GPS, and the Pmod2 compass (see Figures 2.45 and 2.46 for sensor examples). However, each sensor alone does not complete a holistic approach to sensing the robot's surroundings. If the environment changes or the response to the sensors were not exactly repeatable, the robot would end up drastically off course without any ability to recognize this and respond accordingly.

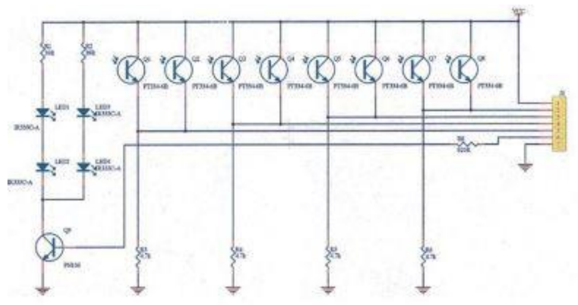
Figure 2.45

IR Sensor

Figure 2.46

IR Sensor Schematic

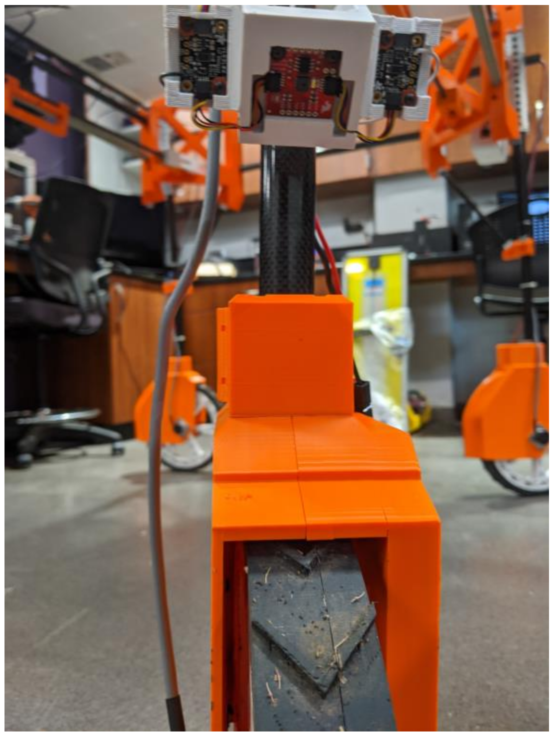




The first sensor developed to determine if the robot could detect wheat was a combination of light emitters and light detectors developed at Langston University, discussed in a previous section. It was proposed that a light-detecting sensor—to determine the amount of light coming in at a particular moment—could be used to calibrate and respond to changing lighting conditions, called the wavelength detector, also previously discussed. However, with the added complexity of the system, a camera would also be able to detect the color of the incoming wavelength and would automatically account for the change in amount of incoming light (see Figure 2.47).

Figure 2.47

Visible Spectrum Sensor with Auxiliary Distance Detectors



The proposed change in sensing equipment to a camera led to image processing techniques. If the robot could filter the pixels for the various colors, then the robot could more effectively sense its surroundings. This is primarily because the ground will usually show as brown, the foliage will be green, and a fully grown harvestable wheat will show as yellow. However, it does have its limitations. If the ground in between the rows is covered in grass or weed, these are too similar of a green for the image processor to distinguish from the wheat. The algorithm would detect many false positives of wheat on a normal pass.

2.14.3 Object Detection Protocol

To account for false positive image processing, a more robust technique is required. The robot must differentiate between grass and wheat because they both reflect similar wavelengths. To accomplish this differentiation, an advanced python object detection program was written. The initial algorithm was modified from a pretrained model using TensorFlow.

In deep learning object detection neural nets, the main process for teaching an algorithm is to retrain a model that has previously been trained on common images. Each of the major AI companies such as Google, Amazon, and Facebook have helped to sponsor the growth of these pretrained models. The two most common of these models are YOLO (You Only Look Once) and COCO (Common Objects in Common Places). They are based on things that people view every day such as a phone, television, people, or pets. These models are designed as pretrained models to be retrained for other uses. By using multiple deep layers of neural nodes, the object detection algorithms have become very accurate at detecting objects that have many labeled images. These deep neural nets have become so accurate that they are better than humans at identifying specifically trained objects and can do so much faster.

The process of retraining these models consists of first training the model in the original method. Then the model is given several new pictures for the new objects to be trained on. The more images the model receives, the more accurate the model will be. Each image contains boundary boxes, and the boxes are labeled with each object's name. The model is then looped through epochs repeatedly, retraining model weights that help detect the particular object more quickly and accurately. This method can be used on images with one labeled object or with multiple different objects if each object has its own boundary box. The boundary boxes can even overlap in the images with objects that are near each other, such as a person holding a cat. For the purposes of the EPSCoR project, the primary goal is to have a working model that detects wheat. However, the model can be retrained to include multiple objects and their corresponding labels. The objects will most definitely be overlapping or have identical boundary boxes. In the case of detecting wheat diseases, the boundary box will be around the wheat as it is detected as wheat and again for the detection of the particular disease. Anything further than wheat detection in this paper is theoretical extrapolation of further techniques that could be beneficial in the field.

The EPSCoR project began the object detection modeling by retraining the YOLO model using 3,400 images of wheat. The wheat was primarily photographed when the stalks had heads but were still green, as this consists of most of the growing cycle. Using the 3,400 images, the model is more than 50% accurate at detecting wheat and can work at speeds between 2 frames per second (FPS) and 7 FPS. If the robot were to drive at about walking speed—approximately 2 mph—the YOLO-based model would be able to read a frame for wheat every 6 in to 20 in. This is not fast enough to respond to the environment. The reason why the YOLO does not achieve a faster frame rate is because of the large number of objects within each image. The YOLO requires more time to process each individual box. A possible solution to this problem would be to have the robot drive slower. However, to achieve the desired results, the robot would need to move about 10 times slower on average, or 0.2 mph. While technically possible, this would be to slow and would affect the functionality of the project.

Alternatively, a faster model than the YOLO could be retrained for our purposes. The Mobileye SSD RCNN model is much faster than the YOLO because it is not affected by large numbers of detected objects within each image and therefore can be more accurate at the desired speed. To use the Mobileye algorithm, the data needed to be converted to the right format. The images originally used a JSON protocol. The Mobileye requires the data to be reformatted into .blob file format. Once the Mobileye model was retrained, the object detection algorithm for wheat was able to reach 24 FPS. This would amount to about 1 frame for every inch or two. In this time frame, the robot will have adequate time to respond to the changing environment around it. Additionally, as the robot drives through the field, the camera will record more images of wheat that can be further used to retrain the model to be more and more accurate over time.

Additional uses of the object detection became readily apparent after the initial phase of the algorithm. The model also could be used to sense the overall health and growth of the wheat; it could be trained to spot diseases and pests; and it could monitor plant height throughout the season. Though this type of analysis was not original to the project, the more data collected on the research wheat, the more robust the genotypes can be predicted.

2.14.4 Object Distance Detection

The model that had been designed up to this point was able to accurately detect if wheat is found within the camera frame. It then could communicate to the robot that it is about to run over wheat, and the robot could respond. However, with this information alone, the robot would not know how exactly to respond. The robot needs to be able to determine whether to turn left, right, or stop completely to avoid running over wheat. To accomplish this, the onboard camera is upgraded from 1MP to 12MP and incorporates two stereo cameras from the OpenCV AI kit (see Figure 2.48).

Figure 2.48

OpenCV AI Camera

The OpenCV AI camera is mounted to one wheel to observe the path the wheel takes within a lane. As the camera detects wheat within the field, the algorithm detects where the wheat is relative to the robot—how far away it is—and then communicates this data to the robot. One method filters the incoming data to wheat detected within a predetermined buffered distance from the robot. If any wheat is detected with a distance from the robot shorter than the predetermined distance, a command is sent to the robot to stop, idle, and investigate. New data can be taken over a longer period to determine if the robot needs to turn, how far away an obstacle is, or simply wait for human instruction. In this instance, the more data collected from all sensors the better. As the robot is stationary, it can compare data from the image-processing algorithm, the wavelength sensor, and the distance sensor in an attempt to obtain a more accurate picture of the robot's immediate surroundings.

Figure 2.49

OpenCV AI Camera Mount

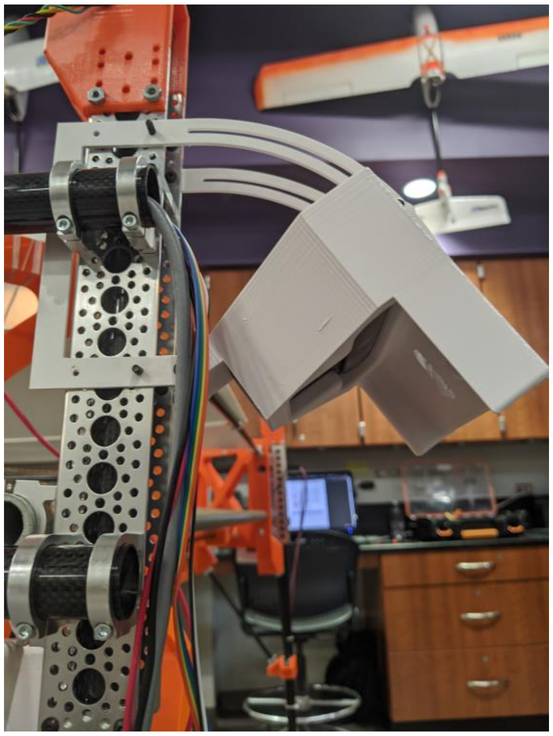


Figure 2.49 is a picture of the mounting system used to attach the camera to the robot. The camera is mounted close to the top of the shaft, nearest to the servos as possible. This allows the field of vision to be its largest while also maintain a fixed distance from the ground. The angle of the camera can then be adjusted from pointed straight at the ground to view the wheel to perpendicular to the ground, allowing the camera to observe more of the path ahead.

2.14.5 Object Detection Path Planning

The camera's sensing abilities give the robot a better field of vision for its surroundings. The robot can now detect where the wheat is, it can learn where the path between the row is, and it potentially can be used to detect diseased wheat or even monitor the wheat height throughout the growing season. Each of these data points can be used by the robot to navigate through the wheat field, augmenting the RTK GPS path.

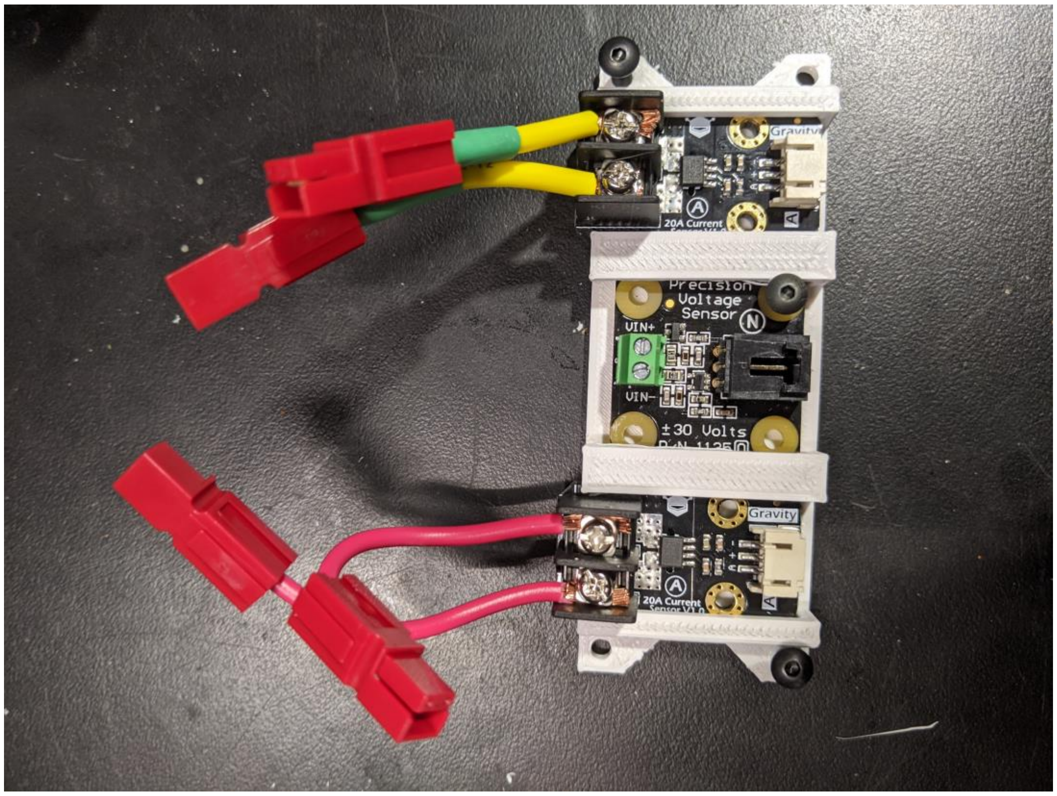
As the robot follows the waypoints specified by the RTK GPS, the cameras can act as quality assurance. If the path is not straight or the robot doesn't track the path correctly, the cameras will be able to detect that the wheel is approaching the wheat. The LattePanda will then send a command for the robot to stop and assess how to proceed. If the robot turned left into the wheat row, then the robot will be able to detect that the wheat was on the left side and the GPS path ran too close to the left edge. The robot will then turn its wheels to the right to shift away from the wheat. It will then save a new GPS waypoint at the specified location to indicate that the robot stopped and in what direction the obstacle was detected. The newly created waypoints that indicate detection are saved as a CSV file on the LattePanda hard drive. These points are later used to compare with the mission planned GPS path. The mission waypoints are then adjusted accordingly. The idea behind this is that the robot will have a better path to follow each time it drives through the field.

In future iterations, the robot could pull up the mission waypoints of the field, find the waypoint closest to the stopping area and alter the waypoint accordingly. In this way, the robot could automatically adjust its mission plan without user input. However, this is outside the scope of this research paper.

2.15 Energy Usage Data

Figure 2.50

Power Meter



To determine the amount of time the robot can reliably run in the field, a power meter for the robot was created (see Figure 2.50). This allows the robot to budget its power accordingly by measuring the incoming power from the solar panels and the outgoing power from batteries to the robot's motors and electronics. By measuring the net power of the robot over different testing environments, a power budget can be created.

Figure 2.51

Solar Power Generated Under Full Sun

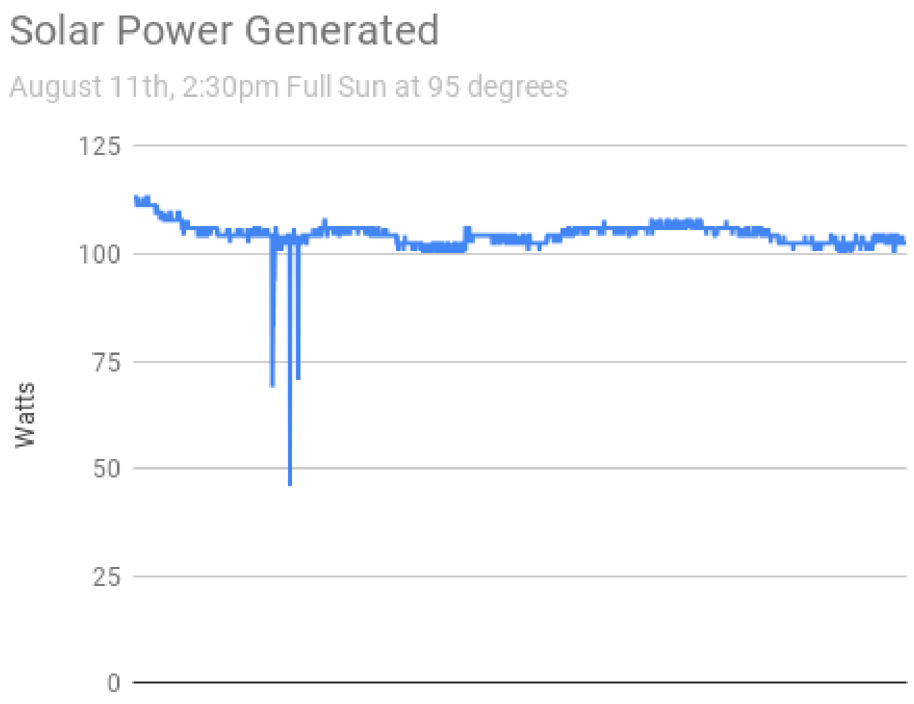
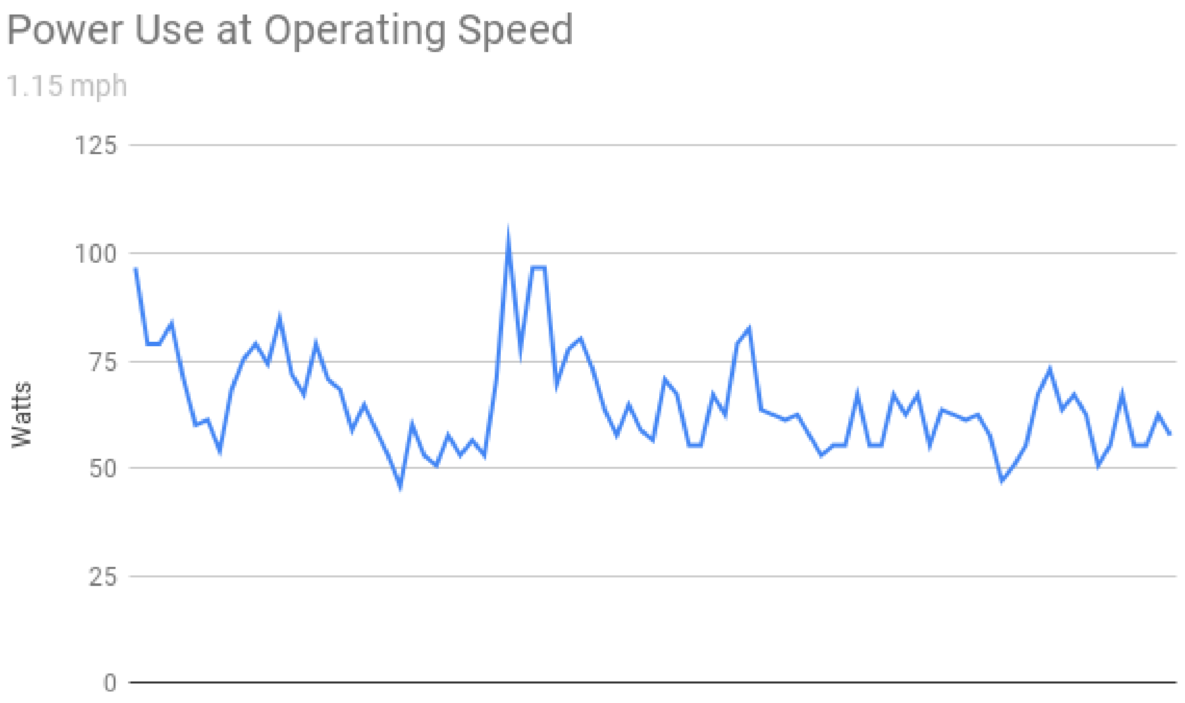


Figure 2.51 shows the incoming power from the solar panel under full sun conditions in the heat of summer in Manhattan, Kansas. The solar charge controller takes this incoming energy and distributes it to the voltage regulator and the excess power gets diverted to the batteries where it is stored for later use. The goal of the solar panels is to have as much or more incoming power as what the batteries use so that the robot can remain in the field indefinitely. Figure 2.52 shows the power used by the robot during typical operating speeds.

Figure 2.52

Robot Power Usage



The average power use of the system is 65 W at normal operating speeds of 1.15 mph. By comparing the data for incoming solar power and the outflow of power from the batteries, the net power graph is created (see Figure 2.53).

Figure 2.53

Net Power of the Robot

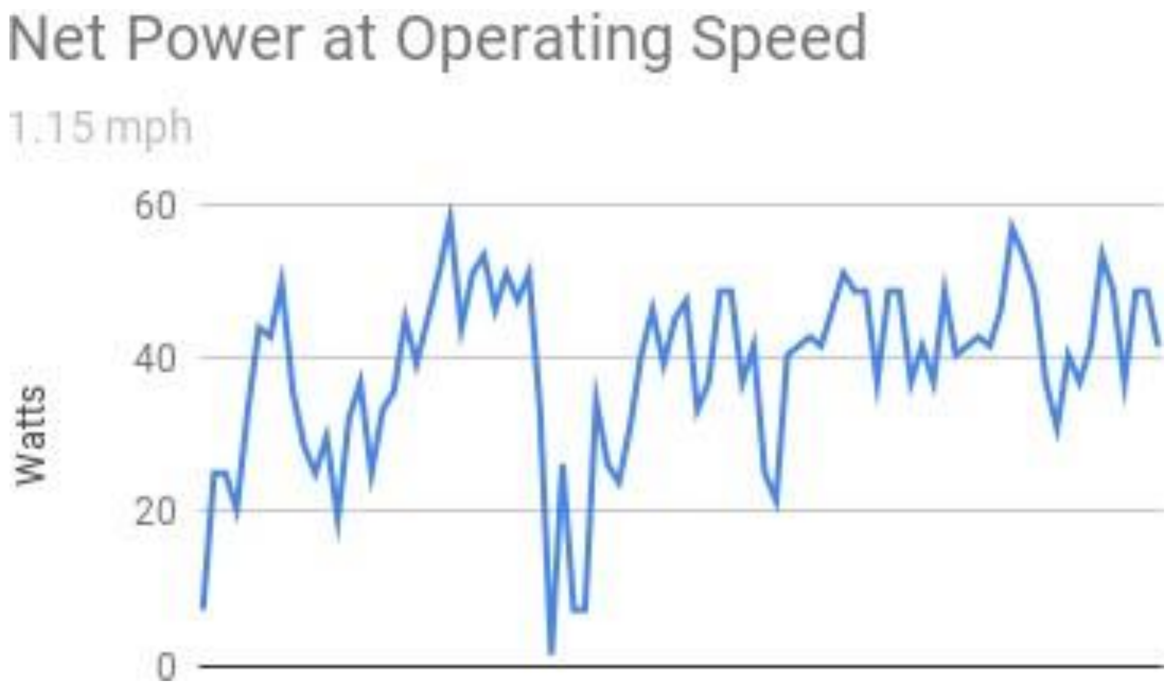
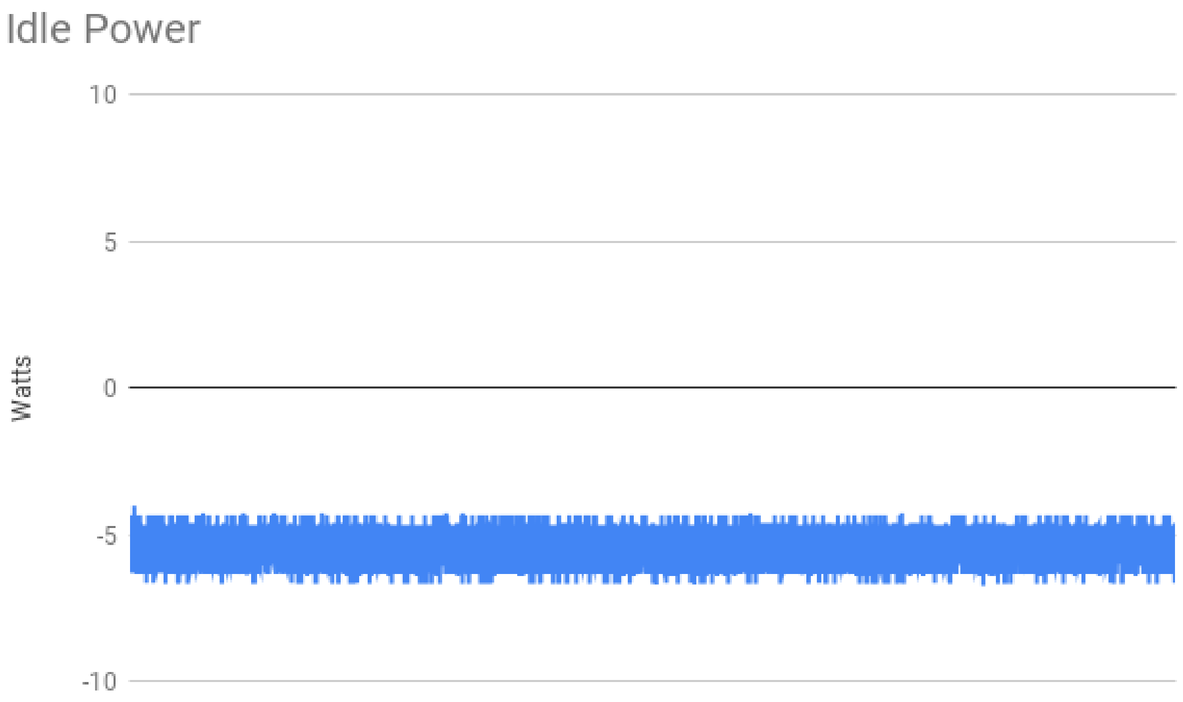


Figure 2.53 shows that the power consistently remains positive throughout typical driving conditions. At certain points, the power in and power out are equivalent, but the power overall consistently remains above zero. This means the batteries will remain full while the robot is in the field, which is especially important because the robot continues to draw power when it is not in use. Figure 2.54 shows that the robot has a consistent draw between 4 W and 6 W of power while the robot is idle.

Figure 2.54

Idle Power Use



The batteries need to have enough remaining energy at the end of the day to sustain the robot through the night, when there is no incoming solar power. At 6 W of loss for 18 hours when the robot is not in full sun, the total energy lost for the system is 108 Wh at most. With a usable battery capacity of 2600 Wh and needing to be at 50% capacity or greater at the start of the day, the solar and battery system will be able to sustain the robot effectively even as the batteries begin to degrade and hold less charge. The state of health of the battery and state of charge for lithium-ion batteries are closely linked to the number of cycles and the aggressive use of the batteries[14]. Because the batteries on the robot will remain above 90% charge at nearly all times, the batteries will experience fewer cycles. When the batteries degrade to half their capacity (past 300 cycles), the incoming power from the solar panels will still be enough to power the robot and retain enough energy through the night when there is no solar power being generated.

The power coming from the solar panels is greater than the power used by the robot under normal conditions, resulting in the batteries never being depleted. This allows for continuous monitoring of the field throughout the growing season.

The same system can be used for a robot that has a net negative power use that will need to change or recharge the batteries. In this circumstance, the power budget can be used to predict when the robot will need to change or recharge its batteries. A system can then be designed to autonomously change the batteries using drones or return to home—features that allow the robot to continuously run in the field without user interference. This would prove useful in a 24-hour planting cycle where the robot can plant for a few hours at a time, switch batteries, and continue on. Normally, the typical in-field run time of machinery is less than 12 hours per day in peak times. But with a system such as this, the robot's effective daily run time could be more than 20 hours.

2.16 Battery Capacity Data

The robot has requirements to operate in the field over the course of the entire growing season. The incoming solar power from the panels will provide 160% of the required power during full sun. But the robot will not be continuously operating and the weather will not provide full sun every day. It is imperative to know the limitations of the robot if there are extended periods where the robot experiences less than full sun or even no sun.

Under cloudy weather, the panel's power decreases by half on normal cloudy conditions and 75% under heavy clouds and rain[1]. Under these conditions, the batteries will maintain full capacity as the robot is in idle. The net power under normal cloudy conditions will be -10 W while in use and -40 W in rainy conditions. In these circumstances, it is necessary to determine if the robot can carry out its determined path in the field for sensing during a day.

A battery capacity assessment was carried out to determine the total usable energy of the system. Figures 2.55 and 2.56 depict the power usage over a 3 hour time frame where the robot drove for 4 miles.

Figure 2.55

3 Hour Power Consumption

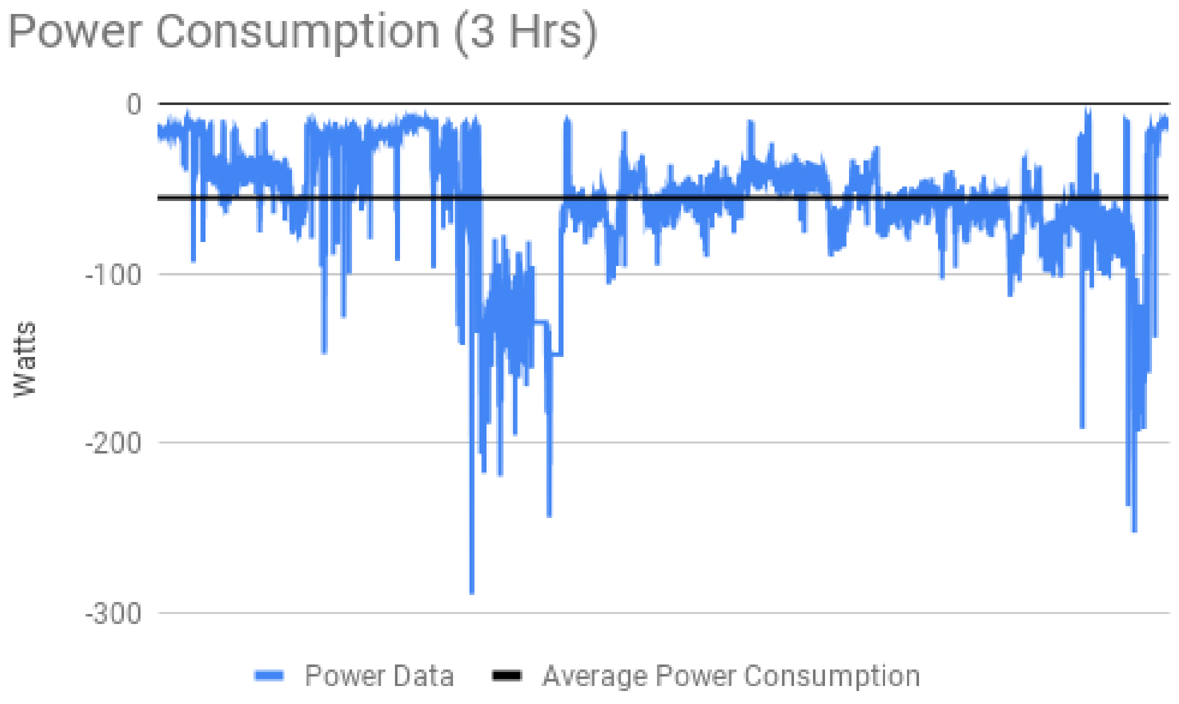
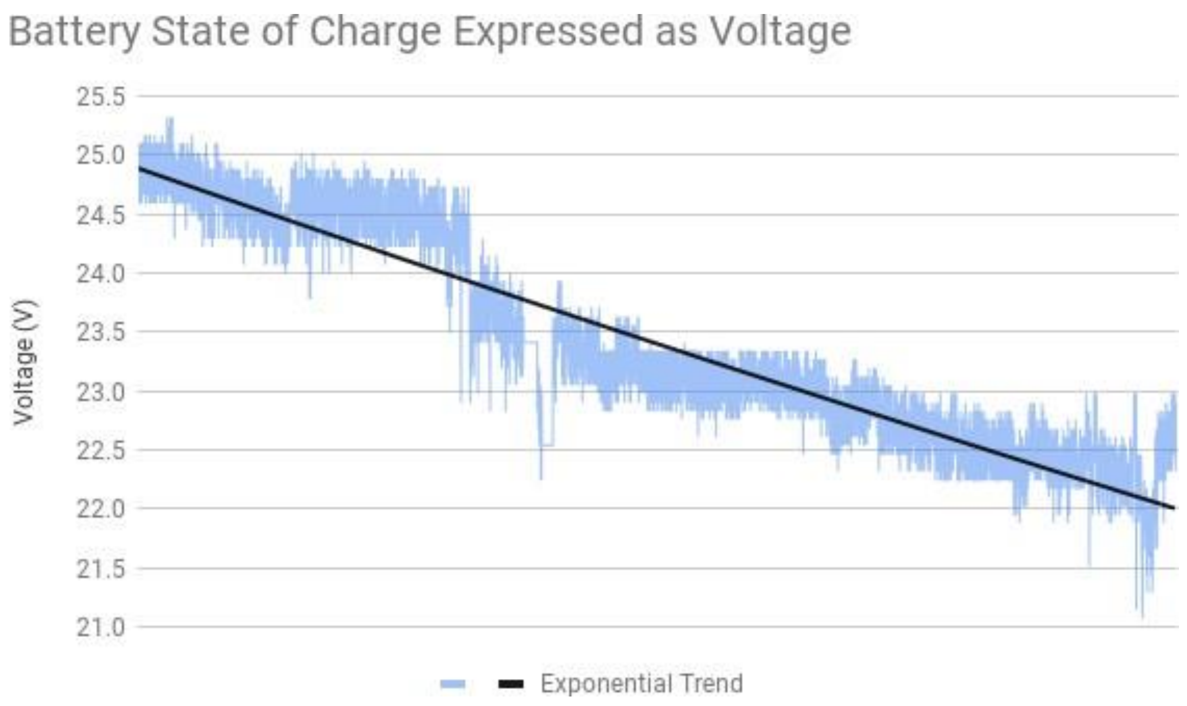


Figure 2.56

Operating Voltage Levels



The robot's total battery capacity is calculated by taking the integral of the power usage over time. The calculated value of 196 Wh of total energy calculated is slightly under the rated capacity of the batteries, which is 235 Wh. State of charge can be monitored using an exponential curve that fits the voltage data.[22] As the observed voltages begin to drop below bins, the voltage can be expressed as a percentage. To maintain the battery health, the state of charge should not drop below 20 percent capacity, giving a usable amount of energy of 157 Wh. This equates to 2.4 hours of battery life at normal operating speeds and 3.25 miles of operating range.

However, there will never be a time when the solar power is at zero during the day. Under the scenarios described above, the Table 2.1 is shown below.
Kansas is under near full sun conditions for 80% of summer days. Under full sun conditions, the robot can theoretically operate indefinitely; the solar panels will continue to charge the battery during the day, even during operation. For the days that are cloudy, the robot

Table 2.1

Battery Expectancy based on real world tests for 196.66 Wh of Capacity

Condition	Solar	Idle	Operation	Operational Time	Distance
Full Sun	104 W	98 W	38 W	-	-
Cloudy	52 W	46 W	-14 W	11.23 hr	15.16 mi
Rain	26 W	20 W	-40 W	3.93 hr	5.31 mi
No Solar	0 W	-6 W	-66 W	2.38 hr	3.22 mi

can operate up to 15 mi of range, more than enough to adequately cover 45 acres in the field. Rainy conditions still provide the robot with adequate battery capacity for a large amount of field tests, however, it is not expected that any tests will be conducted in these conditions. The same is true for times of no solar, such as evenings. The robot can maintain adequate power until the next day's tests.

Chapter 3

Conclusion and Future Work

The unmanned ground vehicle (UGV) project proved to successfully be able to traverse through the research plots of wheat without harming the growth of the wheat. Success was achieved in designing a robot with wheels that could successfully straddle the wheat plot, wheel hubs that followed the intra-row paths without damaging the crop, and enough maneuverability for the robot to remain on course. The UGV also was able to carry the sensor, record data, and save the data to a hard drive for later analysis. The data retrieved from the robot was repeatable, as shown in the testing figures where different driving lines overlapped throughout the course of each testing day.

The high-density lithium polymer batteries and 100 W solar panels give the robot the capability to traverse the entirety of the field. The capacity of the batteries along with the low-drip loss proves the robot's ability to remain in the field without needing to recharge externally. It also proves the battery capacity without solar is enough to permit the UGV to traverse the entirety of the field on days where the solar power is not dependable. In case of a shortfall in energy, the batteries also have a battery protection board to maintain battery health and longevity of the robot operation.

Various sensors on the robot show promise in the robot's capability to drive through the field with little to no human input. The distance sensor, GPS, and wavelength detector each give the robot dependable data sources for line tracking in the field. Additionally, the object detection algorithm created to detect wheat proved successful in both recognizing wheat and determining the distance the wheat is from the robot. This proof of concept can further be used to guide the robot through the field relative to its position from the wheat, limiting conditions of the robot running over wheat accidentally.

Further research includes in-field testing of the sensor to calibrate the EM with the robot's interference, continuing to diminish the influence of noise, and increasing the capabilities of the robot's in-field sensors. When the sensor is calibrated as such, then the robot can be used to record data about the wheat throughout the growing season.

Bibliography

[1] Rabani Adamou Abdoulatif Bonkaney, Sa¨ıdou Madougou. Impacts of cloud cover and dust on the performance of photovoltaic module in niamey. 2017. doi: https://doi.org/10.1155/2017/9107502. URL https://www.hindawi.com/journals/jre/2017/9107502/.

[2] Nikos Alexandratos, Jelle Bruinsma, G Boedeker, J Schmidhuber, S Broca, P Shetty, and M Grazia Ottaviani. World agriculture: Towards 2030/2050. interim report. prospects for food, nutrition, agriculture and major commodity groups. 2006.

[3] BBC. The man who helped feed the world, 2021. URL https://www.bbc.com/news/business-47643456. [Online; accessed May 4th, 2021].

[4] Peiyu Cao, Chaoqun Lu, and Zhen Yu. Historical nitrogen fertilizer use in agricultural ecosystems of the contiguous united states during 1850–2015: Application rate, timing, and fertilizer types. *Earth System Science Data*, 10(2):969–984, 2018. doi: 10.5194/ essd-10-969-2018.

[5] DJI. Dji introduces p4 multispectral for precision agriculture and land management, 2021. URL https://www.dji.com/hk-en/newsroom/news/dji-introduces-p4-multispectral-for-precision-agriculture-and-land-management.
[Online; accessed May 4th, 2021].

[6] Food and Agriculture Organization of the United Nations. Contribution of agricultural growth to reduction of poverty, hunger and malnutrition, . URL http://www.fao.org/ 3/i3027e/i3027e00.htm.

[7] Food and Agriculture Organization of the United Nations. Undernourishment around the world, . URL http://www.fao.org/3/i3027e/i3027e00.htm.

[8] Fran¸cois Tardieu Stephen Welch Bruce Walsh Fredvan Eeuwijk Scott Chapman Dean Podlich Graeme Hammer, Mark Cooper. Models for navigating biological complexity in breeding improved crop plants. *Trends in Plant Sci- ence*, 11, 2006. ISSN 1360-1385. doi: 10.1016/j.tplants.2006.10.006. URL https://www.sciencedirect.com/science/article/pii/S1360138506002810?casa_token=Ois7M1bD4WcAAAAA:0MYZdPA0tgbk8ddaJXEgchmIaHyhcR4arC4VQ_m2hEP7au2AWcD_UOJ0xNBNEYZakgGa5iLw-IM.

[9] Ulrike Grote, Anja Fasse, Trung Thanh Nguyen, and Olaf Erenstein. Food security and the dynamics of wheat and maize value chains in africa and asia, February 2021. URL https://www.frontiersin.org/articles/10.3389/fsufs.2020.617009/full.

[10] I. J. Won Haoping Huang. Real-time resistivity sounding using a hand-held broadband electromagnetic sensor. *Society of Exploration Geophysicists*, 2003.

[11] George R.A. Fields Lynn C. Sutton I.J. Won, Dean A. Keiswetter. Gem-2: A new multifrequency electromagnetic sensor. *Journal of Environmental Engineering Geophysics*, 1, 1996.

[12] John Deere. John deere tractors, 2021. URL https://www.deere.com/sub-saharan/en/tractors/. [Online; accessed May 4th, 2021].

[13] Z. Bertrand Yaneer Bar-Yam Marco LAgi, Karla. Food prices and political instability in north africa and the middle east. 2011.

[14] Corrado Giammanco Stefano Cordiner Aldo Di Carlo Matteo Galeotti, Lucio Cina. Performance analysis and soh (state of health) evaluation of lithium polymer batteries through electrochemical impedance spectroscopy. 2015. ISSN 0360-5442. doi: 10.1016/

j.energy.2015.05.148. URL https://www.sciencedirect.com/science/article/abs/pii/S0360544215007756.

[15] United Nations. World population prospects: The 2015 revision, key findings and advance tables. *Department of Economic and Social Affairs*, 2015. Paper no. ESA/P/WP.241. UN.

[16] Robert L Nielson. Historical corn grain yields in the u.s., Aug 2021. URL https://www.agry.purdue.edu/ext/corn/news/timeless/YieldTrends.html.

[17] Amber Pariona. What are the world's most important staple foods?, Jun 2019. URL https://www.worldatlas.com/articles/most-important-staple-foods-in-the-world.html.

[18] Mohammad Pourkheirandish, Agnieszka A. Golicz, Prem L. Bhalla, and Mohan B. Singh. Global role of crop genomics in the face of climate change. *Frontiers in Plant Science*, 0, 2020. ISSN 1664-462X. doi: 10.3389/fpls.2020.00922. URL https://www.frontiersin.org/articles/10.3389/fpls.2020.00922/full.

[19] World Food Programme. Food insecurity and violent conflict: Causes, consequences, and addressing the challenges. URL https://www.wfp.org/publications/occasional-paper-24-food-insecurity-and-violent-conflict-causes-consequences-and-ad

[20] James I. Sams Bruce D. Smith Garret Veloski Richard Hammack, Burke J. Minsley. Geophysical characterization and monitoring of subsurface drip irrigation, powder river basin, wyoming, usa. *Advanced Systems Enginnering Group*, 2010.

[21] Hannah Ritchie and Max Roser. Crop yields. *Our World in Data*, 2013. https://ourworldindata.org/crop-yields.

[22] Baohua Li Libo Cao Yongzhi Lai Weiwei Zheng Huawen Wang Wei Wang Ruifeng Zhang, Bizhong Xia and Mingwang Wang. A study on the open circuit voltage

and state of charge characterization of high capacity lithium-ion battery under different temperature. 2018. doi: 10.3390/en11092408.

[23] IPPC Secretariat. *Scientific review of the impact of climate change on plant pests - A global challenge to prevent and mitigate plant-pest risks in agriculture, forestry and ecosystems*. FAO on behalf of the IPPC Secretariat, 2021. ISBN 978-92-5-134435-4.

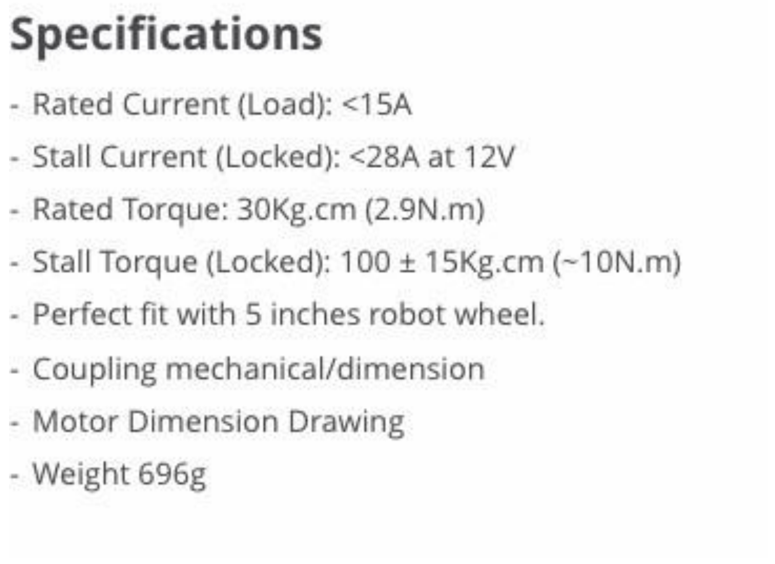
[24] Synbiobeta. Arbor biosciences partners with curio genomics for anal- ysis of iwgsc wheat exome, 2021. URL https://synbiobeta.com/arbor-biosciences-partners-with-curio-genomics-for-analysis-of-iwgsc-wheat-exome/. [Online; accessed May 4th, 2021].

[25] USDA. Farms and land in farms: 2020 summary, 2021. URL https://www.ers.usda.gov/data-products/chart-gallery/gallery/chart-detail/?chartId=58268.

[26] Thom Weir. Farmers Edge, Mar 2021. URL https://www.farmersedge.ca/ effectively-apply-fertilizer-spring/.

Appendix A

Robot Component Specifications



Specifications

- Rated Current (Load): <15A
- Stall Current (Locked): <28A at 12V
- Rated Torque: 30Kg.cm (2.9N.m)
- Stall Torque (Locked): 100 ± 15Kg.cm (~10N.m)
- Perfect fit with 5 inches robot wheel.
- Coupling mechanical/dimension
- Motor Dimension Drawing
- Weight 696g

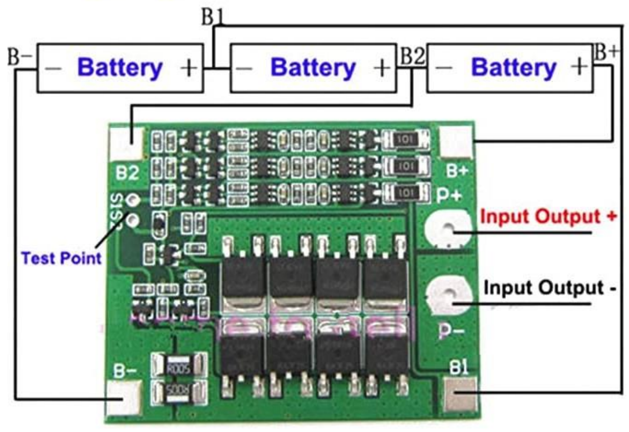
Specifications

- Nominal Voltage (recommended): 12 V (6 V min, 12.6 V max)
- Torque, max. static at 12 V: 29 kg-cm
- Torque, max dynamic at 12 V: 5.8 kg-cm
- Speed, no load at 12 V: 60 RPM
- Current, no load, standby at 12 V: 70 mA
- Current, stall at 12 V: 800 mA
- Operating temperature range: 45°C~85°C
- Spline: 24 Teeth (compatible with Hitec standard spline)
- DC Motor: Coreless
- Gear ratio: 1:320
- Gear material: Steel
- Operating angle: 360° absolute and virtual multi-turn
- Communication type: TTL full duplex asynchronous serial or RC PWM.
- Communication protocol: Custom Lynxmotion Smart Servo (LSS)
- Baudrate range: 9600 bps ~ 921600 bps
- Baudrate (recommended) : 115200bps
- Connector: Molex 4-pin, 2.54mm Low Profile (pinout: Rx | Vcc | GND | Tx)
- Weight: 63 g

What's Included

- **Battery Type:** Lithium Polymer (LiPo Battery)
- **C Rate:** 20C
- **Volts:** 11.1
- **Capacity:** 1300mAh
- **Cell Count:** 3S
- **Cell Configuration:** 3S1P
- **Continuous Discharge:** 20C (26A)
- **Max Burst Rate:** 30C (39A)
- **Max Volts per Pack:** 12.6V
- **Min Volts per Pack:** 9V
- **Charge Rate:** 1C (1.3A)
- **Wire Gauge:** 14 AWG Soft and Flexible Low Resistance Silicone Wire
- **Plug Type:** Venom UNI 2.0 Land Plug System. Compatible with XT60 Plug, Traxxas® Plugs, Deans Plug & EC3 Plug.
- **Dimensions:** 96.5 x 34 x 19 mm / 3.8 x 1.3 x .7 in
- **Watt Hours:** 14.43

Wiring Diagram:



P+: Input Output +
P- : Input Output -

Specification:
Over voltage range: 4.25 4.35v¡À0.05v
Over discharge voltage range: 2.3 3.0v¡À0.05v
Maximum operating current: 0 25A
Maximum transient current: 34 40A
Quiescent current: less than 30uA
Internal resistance: less than 100m?
Working temperature: 40 +50?
Storage condition: 40 +80?
Effective life: more than30000h
Short circuit protection: Yes, delayed self recovery
Size: 56mm*45mm*1.2mm(LXWXH)
Material: PCB
Color: Green
Quantity: 1 PC
Net Weight: 12g

B125 GNSS OEM Board

TRACKING	
Channels	226 Universal Tracking Channels™
Signals Tracked	GPS: L1, L2, L2C, L5 GLONASS: L1, L2, L3 Galileo: E1, E5a, E5b, E5AltBOC BeiDou: B1, B2 QZSS: L1, L2, L1C, L1-SAIF, L2C, L5 SBAS: L1 L-Band
ACCURACY[1] (RMS)	
Standalone	H: 1.2 m; V: 1.8 m
DGPS	H: 0.3 m; V: 0.5 m
SBAS	H: 0.8 m; V: 1.2 m
RTK	H: 5 mm + 0.5 ppm x baseline; V: 10 mm + 0.8 ppm x baseline
RTK Initialization	Time: < 10 seconds Reliability: > 99%
HD2	Heading 0.2°/D, where D is the inter-antenna distance in meters Inclination 0.3°/D, where D is the inter-antenna distance in meters
Velocity	0.02 m/second
Time	30 nsec
ACQUISITION TIME	
Hot / Cold Start	< 15 sec / < 44 sec typical
Reacquisition	< 1 sec
COMMUNICATION INTERFACES	
RS232	2x ports up to 460.8 kbps
LVTTL UART	1x ports up to 460.8 kbps
USB 2.0 (client)	1x port up to 480 mbps (High Speed)
CAN	1x port (without transceivers), CAN 2.0 A/B , NMEA2000 compliant
Ethernet	1x port supporting TCP/IP, FTP, Ntrip Server/Client

I/O	
PPS	1x output with 5 ns resolution, LVTTL, configurable edge, period, offset, and reference time
EVENT	1x input with 5 ns resolution, LVTTL, configurable edge and reference time
DATA AND MEMORY	
SD card support	Industrial SLC SD card, 20 Hz writing rate, up to 32GB capacity
Data Update/Output Rate	1 Hz – 100 Hz Selectable
Data Formats	TPS, RTCM SC104 2.x and 3.x, CMR/CMR+[2], BINEX
ASCII Output	NMEA 0183 versions 2.x, 3.x, and 4.x
ENVIRONMENTAL	
Temperature	Operating: -40°C to 85°C; Storage: -40°C to 85°C
Vibration	4g Sine Vibe (SAEJ1211); 7.7g Random Vibe (MIL-STD 810F)
Humidity	95%, non-condensing
Shock	Operational IEC68-2-27, 11ms, 40g Survival IEC68-2-27, 11ms, 75g
Acceleration	20g
POWER	
Voltage / Power Consumption	3.4 VDC to 5.5 VDC / 2.0 W typical
LNA Power	+3.4 V to +5.5 V (internal), +4.8 V to +5.16 V (external) at 0 – 120 mA
PHYSICAL	
Dimensions / Weight	55 mm x 40 mm x 10 mm / 20 g
Main Connector	80-pin Hirose
Antenna Inputs	2 ESD protected
Antenna Connectors	Hirose H.FL

1. These specifications will vary depending on the number of satellites used, obstructions, satellite geometry (PDOP), occupation time, multipath effects, and atmospheric conditions. Performance may be degraded in conditions with high ionospheric activity, extreme multipath, or under dense foliage. For maximum system accuracy, always follow best practices for GNSS data collection.

2. CMR/CMR+ is a third-party proprietary format. Use of this format is not recommended and performance cannot be guaranteed. Use of industry standard RTCM 3.x is always recommended for optimal performance.

Appendix B

Code for the Pancreas Robot

B.1 Code for the MyRio

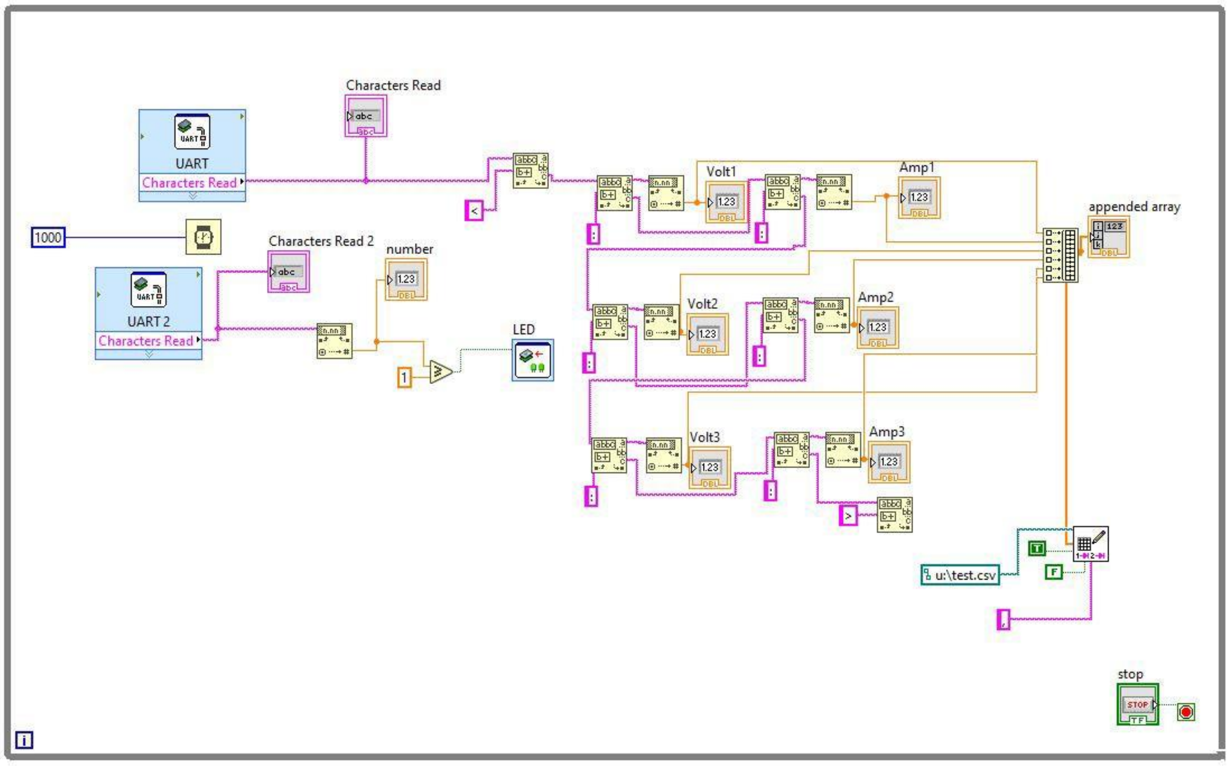


Figure B.1: LabVIEW Communication Code

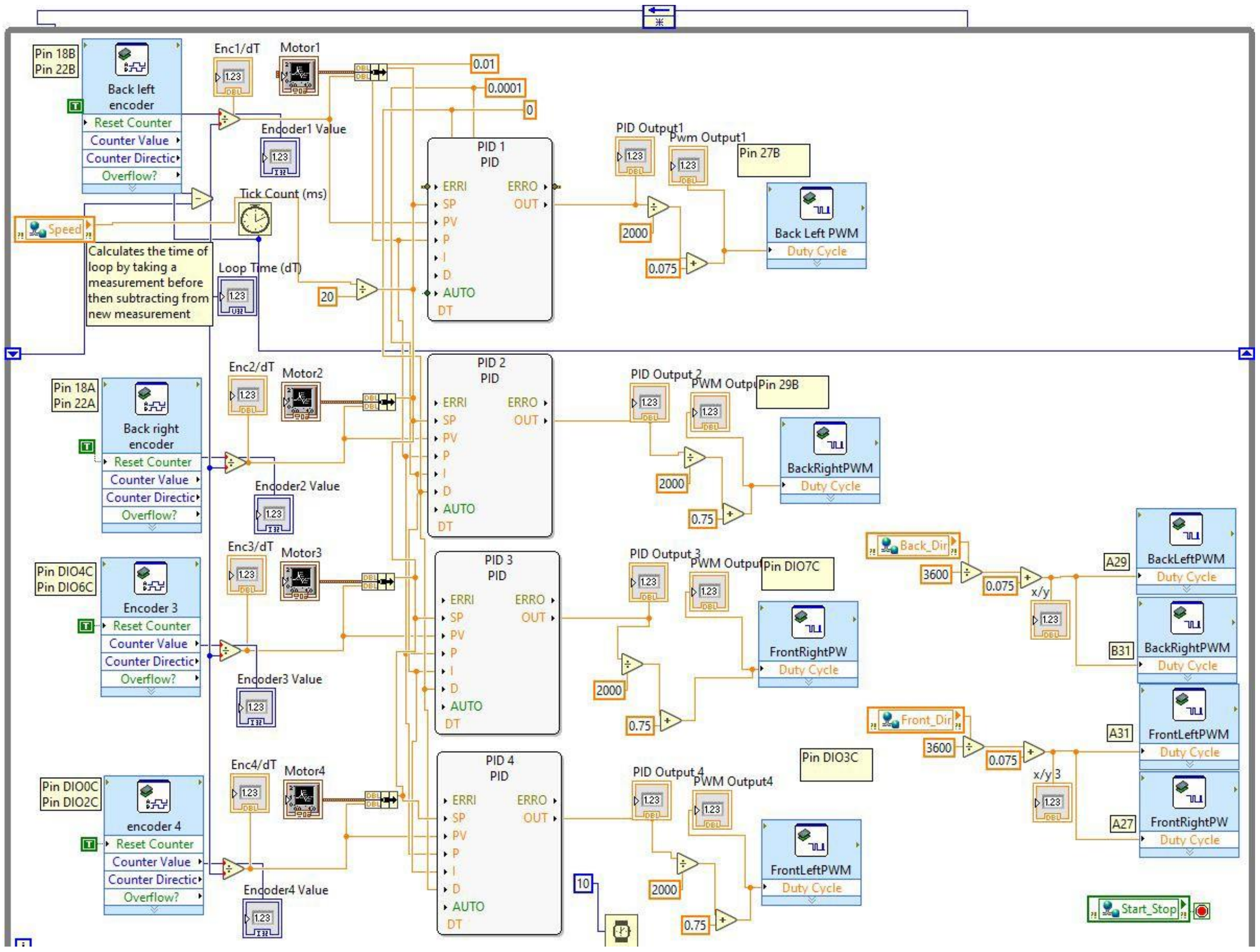


Figure B.2: LabVIEW PID Code

B.2 LattePanda

# -*- coding: utf-8 -*-

#!/usr/bin/python

import serial import

time import

pynmea2 import

math

```
from numpy import genfromtxt import
re
import csv
#from gps import *
#import numpy as np

#initializes waypoints

wp_num = 0
wp_heading = 0
wp_lat  =  39.19024606333333
wp_lon  =  -96.58274821833334
gps_lat  =  39.190277
gps_lon  =  -96.588887

wp2 = genfromtxt(r’ C:\Users\LattePanda\Desktop\wp.csv’ , delimiter=’ ,’ ) print(wp2)



#Arduino Serial Port
# arduino = serial.Serial(port =’ COM6’ ,baudrate = 115200,timeout=5) # time.sleep(2) #
wait for Arduino
#GPS Serial Port
# gps = serial.Serial(port =’ COM4’ ,baudrate = 115200,timeout=5) # time.sleep(2)
# wait for GPS

# print ("Is port open ? ", gps.isOpen())
```

```
def write_read(x): #communucation btw the cpu and ard
arduino = serial.Serial(port =’ COM6’ ,baudrate = 115200,timeout=5)
arduino.write(bytes(x,’ utf-8’ ))
arduino.flushInput() #flush input buffer, discarding all its contents arduino.flushOutput()
time.sleep(.05)
data = arduino.readline()
arduino.close()
return data

def write_toArduino(x): #communucation btw the cpu and ard
arduino = serial.Serial(port =’ COM6’ ,baudrate = 115200,timeout=5)
arduino.write(bytes(x,’ utf-8’ ))
arduino.flushInput() #flush input buffer, discarding all its contents arduino.flushOutput()
time.sleep(.05)
arduino.close() return

def read_fromArduino(): #communucation btw the cpu and ard
arduino = serial.Serial(port =’ COM6’ ,baudrate = 115200,timeout=5) arduino.flushInput() #flush
input buffer, discarding all its contents arduino.flushOutput()
time.sleep(.05)
data  =  arduino.readline()
```

```
arduino.close() return data

def read_gps(): #Reads the GPS Serial Port for NMEA data = ""

gps = serial.Serial(port =’ COM8’ ,baudrate = 115200,timeout=5) #time.sleep(2) #
wait for GPS
gps.flushInput() #flush input buffer, discarding all its contents gps.flushOutput()
time.sleep(.05)
data = gps.readline()
time.sleep(.05) gps.close()

return data

def get_waypoint(waypoint_num): #updatest the waypoint return
waypoint_num + 1

#calculates distance from target location
def distance_from_waypoint(lat1, lon1, lat2, lon2):
R = 6371000 #Earths Radius in Meters dlat =
deg2rad(lat2-lat1)
dlon  =  deg2rad(lon2-lon1)
a      =      math.sin(dlat/2)*math.sin(dlat/2)+math.cos(deg2rad(lat1))*math.cos(deg2rad(lat2) c = 2 *
math.atan2(math.sqrt(a), math.sqrt(1-a))
```

```
x = int(R * c *10000)
#      distance_to_waypoint      =      R      *      c
distance_to_waypoint      =      x/10000      return
distance_to_waypoint*100

#calculates desired heading
def heading_to_waypoint(lat1, lon1, lat2, lon2): wp_heading =
math.atan2(lat2-lat1, lon2-lon1) wp_heading *= (180/math.pi)
wp_heading = int((450-wp_heading)%360) print("WP Heading")
print(wp_heading) return
wp_heading

def deg2rad(deg):
return deg*(math.pi/180)

#  def  getPositionData(value):
#            nx  =  gpsd.next()

#            if nx[’  class’  ] == ’  TPV’  :
#                      latitude =  getattr(nx, ’  lat’  ,  "Unknown") #
                       longitude = getattr(nx, ’  lon’  , "Unknown")
#                      print ("Your position: lon = " + str(longitude) + ", lat = " + str(latitude))

def export_to_csv(wp_lat, wp_lon, wp_heading, gps_lat, gps_lon, compass, distance, solar message = [wp_lat, wp_lon,
wp_heading, gps_lat, gps_lon, compass, distance, solar_po
```

```
print (message)

with open(r’ C:\Users\LattePanda\Desktop\data.csv’ ,’ a’ ,newline=’ ’ ) as f: # using csv.writer

method from CSV package
write = csv.writer(f) write.writerow(message)

return



while True: try:
gps_data = read_gps() #print(value)
gps_data = gps_data.decode("utf-8") #print(gps_data)
gps_data_parse = pynmea2.parse(gps_data) print(gps_data_parse)
gps_lat = gps_data_parse.latitude gps_lon =
gps_data_parse.longitude

except:
print("No connection to GPS") pass
```

```
position = str(gps_lat) + "," + str(gps_lon) #Gives robots current
GPS location print("Robot Coordinates")
print(position)

# position2  =  write_read(position).decode("utf-8") # print(position2)
#Calculates robots current distance to waypoint
distance = distance_from_waypoint(gps_lat, gps_lon, wp_lat, wp_lon) #calculates robots desired
heading
wp_heading = heading_to_waypoint(gps_lat, gps_lon, wp_lat, wp_lon)

#Chooses next wp after robot reaches previous wp while distance < 1:
print(wp_num) wp_num += 1
wp_lat = wp2[wp_num,0] wp_lon =
wp2[wp_num,1]
distance = distance_from_waypoint(gps_lat, gps_lon, wp_lat, wp_lon) print(distance)
#Gives robots distance to WP print("Distance to
Waypoint (cm)") print(distance)

time.sleep(.05) #send to

arduino
```

```
#format <desired_heading, distance_to_waypoint, on/off> robot_on_off = 1
communication = ""
communication = "<" + str(wp_heading) + "," + str(distance) + "," + str(robot_on_off #print(communication)
compass = "" try:
arduino_communication = ""; arduino_communication =
read_fromArduino()
arduino_communication = arduino_communication.decode("utf-8") comm2 =
arduino_communication[1:-3]
comm3 = ""
comm3 = (re.split(' , ' , comm2))
#print(arduino_communication) solar_power =
comm3[0] battery_power =  comm3[1] net_power
=  comm3[2]
compass = comm3[3] compass =
float(compass)
except:
print("Error with Arduino Comm") pass

export_to_csv(wp_lat, wp_lon,  wp_heading,  gps_lat,  gps_lon,  compass,  distance,  solar # communication_return =
write_read(communication).decode("utf-8")
# print(communication_return) # print("")
```

B.3 Code for the Arduino

```
/*********************************************************************** Description:
Pmod_CMPS2
The three components (X, Y, Z) of the magnetic field and then its module and its angle are displayed
in the serial monitor

Wiring
Module<----------> Arduino
    VCC   to        3V3
    GND   to        GND
    SCL   to        A5 (SCL)
    SDA   to        A4 (SDA)
************************************************************************/
// The earth’ s magnetic field varies according to its location.
// Add or subtract a constant to get the right value
// of the magnetic field using the following site
// http://www.ngdc.noaa.gov/geomag-web/#declination
#define DECLINATION -70// Rotates where true north is relative to +X axis compass readin
/************************************************************************/ #include
<Wire.h> //including library for I2C communication
unsigned char CMPS2_address = 0x30; //I2C address of the device const int
currentSensorPin = A2; //define sensor pin
const int voltageSensorPin = A1;
const int currentSensorPin2 = A5; //define sensor pin
const int mVperAmp = 100; // use 185 for 5A Module, and 66 for 30A Module float angle =0;
float Vref          = 0; //read your Vcc voltage,typical voltage should be 5000mV(5.0V) int VoltValue = 0;
                    // value read from the pot
float Voltage = 0;                      // calculated Voltage from sensorValue float Power
= 0;                                    // calculated Voltage from sensorValue float Power2 =
0;                                      // calculated Voltage from sensorValue float
PowerNet = 0;                           // calculated Voltage from sensorValue
//#include <SoftwareSerial.h> //Load the Software Serial Library. This library in effect
//SoftwareSerial mySerial(3, 2); //Initialize SoftwareSerial, and tell it you will be char c;        //Used to read the
characters spewing from the GPS module
String Comm = ""; String
Heading =""; void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
Serial.begin(115200); //serial initialization delay(10);
CMPS2_init(); //initialize the compass delay(10);
                        //Pause
Vref = readVref(); //read the reference votage(default:VCC) delay(10); //Pause
digitalWrite(LED_BUILTIN, HIGH);              // turn the LED on (HIGH is the voltage level) delay(100);
Serial.println("1");
}

float data[3];
```

```
unsigned int raw[3]; float
offset[3];

void loop() {
delay(100);
// wait for a second
//digitalWrite(LED_BUILTIN, LOW);                    // turn the LED off by making the voltage LOW

//retrieving and displaying the heading of the compass float angle =
CMPS2_getHeading();
float  CurrentValue  =                readDCCurrent(currentSensorPin); float
CurrentValue2  =                      readDCCurrent(currentSensorPin2); VoltValue
= analogRead(voltageSensorPin);
Voltage = (VoltValue - 512) * 0.073170;
Power = 24 * CurrentValue; //Power from the batteries Power2 = 36 *
CurrentValue2; //Power from solar
PowerNet = Power2 - Power;//Solar Power minus Power used from Batteries Heading = String(angle);
Comm = "<" + String(CurrentValue2) + "," + String(CurrentValue) + "," + String(Volta Serial.println(Comm);




//          Serial.println(PowerNet);
////        Serial.print("Heading = ");
//          Serial.println(angle);
//        Serial.print("°");
//        Serial.println(’  \t’  );
// For if you want the cardinal directions
//        CMPS2_decodeHeading(angle);          //get direction
}

void CMPS2_decodeHeading(float angle) {
//decoding heading angle according to datasheet
//        if (angle > 337.25 | angle < 22.5) {
//            Serial.println("North");
//        }
//        else if (angle > 292.5) {
//            Serial.println("North-West");
//        }
//        else if (angle > 247.5) {
//            Serial.println("West");
//        }
//        else if (angle > 202.5) {
//            Serial.println("South-West");
//        }
//        else if (angle > 157.5) {
//            Serial.println("South");
//        }
//        else if (angle > 112.5) {
//            Serial.println("South-East");
//        }
//        else if (angle > 67.5) {
```

```
//          Serial.println("East");
//       }
//       else {
//          Serial.println("North-East");
//       }
}



float CMPS2_getHeading(void) {
CMPS2_read_XYZ();          //read  X,  Y,  Z  data  of  the  magnetic  field

//eliminate offset before continuing for (int
i=0;i<3;i++)
{
data[i]  =  data[i]-offset[i];
}

//variables for storing partial results float temp0 = 0;
float temp1 = 0;
//and for storing the final result float deg = 0;

//calculate heading from data of the magnetic field
//the formula is different in each quadrant if (data[0] < 0)
```

```
{
if (data[1] > 0)
{
//Quadrant  1 temp0 = data[1];
temp1 = -data[0];
deg = 90 - atan(temp0 / temp1) * (180 / 3.14159);
}
else
{
//Quadrant  2 temp0 = -data[1];
temp1 = -data[0];
deg = 90 + atan(temp0 / temp1) * (180 / 3.14159);
}
}
else {
if (data[1] < 0)
{
//Quadrant  3 temp0 = -data[1];
temp1 = data[0];
deg = 270 - atan(temp0 / temp1) * (180 / 3.14159);
}
else
{
//Quadrant 4
temp0 = data[1]; temp1 = data[0];
deg = 270 + atan(temp0 / temp1) * (180 / 3.14159);
}
}

//correct   heading deg +=
DECLINATION; if
(DECLINATION > 0)
{
if (deg > 360) { deg -= 360;
}
}
else
{
if (deg < 0) { deg += 360;
}
}

return deg;
}



//reads measurements in mG
```

```
void CMPS2_read_XYZ(void) {
//initialize array for data

//command internal control register 0 bit 0 (measure)
Wire.beginTransmission(CMPS2_address); Wire.write(0x07);
Wire.write(0x01);
Wire.endTransmission(); delay(8);

//wait for measurement to be completed bool flag =
false;
while (!flag) {
//jump to status register
Wire.beginTransmission(CMPS2_address);
Wire.write(0x06); Wire.endTransmission();

//read its value Wire.requestFrom(CMPS2_address, (uint8_t)1);
int temporal = 0;
if (Wire.available()) { temporal =
Wire.read();
}

//if the last bit is 1, data is ready temporal &= 1;
```

```
if (temporal != 0) { flag = true;
}
}

//move address pointer to first address
Wire.beginTransmission(CMPS2_address);
Wire.write(0x00); Wire.endTransmission();

//save data
Wire.requestFrom(CMPS2_address,  (uint8_t)6);
byte tmp[6] = {0, 0, 0, 0, 0, 0}; //array for raw data if (Wire.available()) {
for (int i = 0; i < 6; i++)  { tmp[i] = Wire.read();
//save it
}
}

//reconstruct raw data
raw[0] = tmp[1] << 8 | tmp[0]; //x raw[1] = tmp[3]
<< 8 | tmp[2]; //y raw[2] = tmp[5] << 8 | tmp[4];
//z

//convert raw data to mG
for (int i = 0; i < 3; i++) {
data[i] = 0.48828125 * (float)raw[i];
```

```
}
}

//initialize the compass
//Update:    this    should    follow    Calibration    steps    void
CMPS2_init(void) {
float    out1[3];    float
out2[3]; int i;

Wire.begin(); // initialization of I2C bus

//calibration: SET Wire.beginTransmission(CMPS2_address);
Wire.write(0x07);
Wire.write(0x80);
Wire.endTransmission(); delay(60);

Wire.beginTransmission(CMPS2_address); Wire.write(0x07);
Wire.write(0x20);
Wire.endTransmission(); delay(10);

CMPS2_read_XYZ(); for
(i=0;i<3;i++)
```

```
{
out1[i] = data[i];
}
Serial.print("Raw SET = ");
Serial.print(raw[0]); Serial.print("\t");
Serial.print(raw[1]); Serial.print("\t");
Serial.println(raw[2]);

//calibration: RESET Wire.beginTransmission(CMPS2_address);
Wire.write(0x07);
Wire.write(0x80);
Wire.endTransmission(); delay(60);

Wire.beginTransmission(CMPS2_address); Wire.write(0x07);
Wire.write(0x40);
Wire.endTransmission(); delay(10);

CMPS2_read_XYZ(); for
(i=0;i<3;i++)
{
out2[i] = data[i];
}

Serial.print("Raw RESET = ");
Serial.print(raw[0]); Serial.print("\t");
Serial.print(raw[1]); Serial.print("\t");
Serial.println(raw[2]);

//offset calculation for
(i=0;i<3;i++)
{
offset[i]  =  (out1[i]+out2[i])*0.5;
}

//command internal control register 0 for set operation
Wire.beginTransmission(CMPS2_address); Wire.write(0x07);
Wire.write(0x40);                  //SET
Wire.endTransmission(); delay(10);

//command internal control register 1 to 16 bit resolution, 8ms measurement time
Wire.beginTransmission(CMPS2_address);
Wire.write(0x08);
Wire.write(0x00);
Wire.endTransmission();
```

```
delay(10);
}
float readDCCurrent(int Pin)
{
int analogValueArray[31];
for(int index=0;index<31;index++)
{
analogValueArray[index]=analogRead(Pin);
}
int i,j,tempValue;
for (j = 0; j < 31 - 1; j ++)
{
for (i = 0; i < 31 - 1 - j; i ++)
{
if (analogValueArray[i] > analogValueArray[i + 1])
{
tempValue = analogValueArray[i]; analogValueArray[i] = analogValueArray[i + 1];
analogValueArray[i + 1] = tempValue;
}
}
}
float medianValue = analogValueArray[(31 - 1) / 2];
float DCCurrentValue = (medianValue / 1024.0 * Vref - Vref / 2.0) / mVperAmp;                    //Sen
return DCCurrentValue;
}
```

```
/*read reference voltage*/ long
readVref()
{
long result;
#if defined(_AVR_ATmega168_) || defined(_AVR_ATmega328_) || defined (_AVR_ATmega328 ADMUX =
_BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
#elif defined(_AVR_ATmega32U4_) || defined(_AVR_ATmega1280_) || defined(_AVR_ATmega ADMUX =
_BV(REFS0) | _BV(MUX4) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
ADCSRB &= ~_BV(MUX5);              // Without this the function always returns -1 on the ATmega #elif defined (
AVR_ATtiny24_) || defined(_AVR_ATtiny44_) || defined(_AVR_ATtiny84_
ADMUX = _BV(MUX5) | _BV(MUX0);
#elif defined (_AVR_ATtiny25_) || defined(_AVR_ATtiny45_) || defined(_AVR_ATtiny85_ ADMUX = _BV(MUX3) |
_BV(MUX2);
#endif
#if defined(_AVR_)
delay(2);                                                    // Wait for Vref to settle
ADCSRA |= _BV(ADSC);                                         // Convert
while (bit_is_set(ADCSRA, ADSC));
result  =   ADCL; result |= ADCH
<< 8;
result = 1126400L / result;                 //1100mV*1024 ADC steps http://openenergymonitor.org/em return
result;
#elif defined(_arm_)
return  (3300);                                             //Arduino Due
#else
return (3300);                                              //Guess that other un-supported arch #endif
}
```

B.4 Code for the Object Detection Camera

```
# -*- coding: utf-8 -*- """yolo.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/1evD2ZxDWNGCmeq_6ZNfUZY31XwH-Tfqb
"""

from google.colab import drive drive.mount(' /content/drive' )

# Commented out IPython magic to ensure Python compatibility. # %cd
/content/drive/MyDrive/yolov5

!pip install -r requirements.txt

 # Commented out IPython magic to ensure Python compatibility.
 # %time
 # !python train.py --img 320 --batch 24 --epochs 100 --data wheat.yaml --cfg models/yolo
 # !python train.py --img 320 --batch 24 --epochs 100 --data wheat.yaml --cfg models/yolo

 # Commented out IPython magic to ensure Python compatibility.
# %time
!python train.py --img 640 --workers 16 --batch 8 --epochs 100 --data wheat.yaml --cfg

# !python detect.py --source /Users/sujithgunturu/Downloads/a.jpg --weights /Users/sujit # !python train.py --img 1024 -
-batch 8 --epochs 100 --data wheat.yaml --cfg models/yolo

# !python detect.py --source 0 --weights /Users/sujithgunturu/global-wheat-detection/yol

# Commented out IPython magic to ensure Python compatibility. # %load_ext
tensorboard
# %tensorboard --logdir /content/drive/MyDrive/yolov5/runs/train/wml
```