Robotic Transportation Vehicle

Alex Koschmann
Stevie Baldwin
Haotian Cai
Timur Yaprak
Grace Jones

Facilitator: Fang Peng
Sponsor: Tim Adcock
Executive Summary

The objective of this design project was to develop and configure a functioning robotic transportation prototype vehicle for Texas Instruments. The development of this prototype vehicle involved many different engineering domains including, wireless communication, platform assembly, motor control, movement control, and power distribution. The wireless communication portion of the design involved developing programming code, which would interpret signals sent from the Chronos watch to the Stellaris LaunchPad. The Stellaris LaunchPad, which acted as a central processor, would translate these signals to our motor controllers. This would in turn rotate the motors at certain RPMs. According to the movement diagram, movement of the vehicle should seamlessly reflect the movement and tilt of the watch. This would be accomplished by integrating the motor controllers effectively with the central processor. To power the system, Team 9 had to determine the various power conversions that needed to be made in order to supply the different electrical components sufficient power to operate.

The design team was able to create a working prototype for Texas Instruments that they will be able to utilize in their motor and robotics labs. However, due to electrical damage to one of the components, the vehicle will not be able to move in the backwards direction. Due to this same issue, the team also had to incorporate brushed DC motors, despite the brushless DC motor design specification.

The final report goes through the progression of the design project and highlights the challenges and critical decisions Team 9 had to make throughout the duration of the semester.
Acknowledgements

Design Team nine would like to give special thanks to team sponsor Tim Adcock, Stephen Fedigan, and Texas Instruments’ motor lab for sponsoring the project. We would also like to give thanks to our facilitator Dr. Peng and Professor Shanblatt, for the guidance they provided throughout the semester. Additionally, we would like to thank Roxanne Peacock from the Electrical Engineering Department. She was very helpful with our purchase orders, as well as specific design suggestions that helped keep our budget low. And finally, Team 9 would like to thank Brian Wright and Gregg Mulder for their continuous help in the lab, and providing everyone with the specific parts and equipment that were needed. Without the help of these individuals, Design Team 9 would not have had the resources to finish the design project, and are very thankful for their endless support.
# Table of Contents

Executive Summary .................................................................................................................. 2
Acknowledgements .................................................................................................................. 3
Chapter 1: Introduction and Background .............................................................................. 6-7
  Introduction ......................................................................................................................... 6
  Background ......................................................................................................................... 6
  Design Specifications and Objectives ................................................................................... 7
Chapter 2: Technical Approaches ............................................................................................. 8-16
  FAST Diagram .................................................................................................................... 8
  House of Quality .................................................................................................................. 9
  Critical Customer Requirement (CCR) ............................................................................... 9-10
  Conceptual Design Descriptions ....................................................................................... 10-12
  Ranking of Conceptual Designs ......................................................................................... 13
  Budget ................................................................................................................................. 14
  Gantt Chart ......................................................................................................................... 15
  Project Management ........................................................................................................... 16
Chapter 3: Technical Description of Work Performed ............................................................... 17-20
  Hardware Design Efforts .................................................................................................... 17-18
  Hardware Implementation ................................................................................................. 19
  Software Interface, Design and Implementation ............................................................... 19-20
  Problems Encountered ..................................................................................................... 20-21
Chapter 4: Engineering Data and Results ................................................................................. 22
  Data and Results ................................................................................................................. 22-27
  Final Prototype Description ............................................................................................... 28
Chapter 5 .................................................................................................................................. 29
  Conclusion .......................................................................................................................... 29
Appendices

Appendix 1: Technical Roles, Responsibilities and Work Accomplished .................................................................1
   Team Manager Contribution .................................................................................................................................2
   Webmaster Contribution .........................................................................................................................................3
   Presentation Preperator Contribution .................................................................................................................4
   Documentation Preperator Contribution .............................................................................................................5
   Lab Coordinator Contribution ............................................................................................................................6

Appendix 2: Literature and Website Reference ........................................................................................................7

Appendix 3: Technical Documentation ..................................................................................................................8-9
Chapter 1: Introduction and Background

Introduction

Texas Instruments, a globally renowned company for over eighty years, mainly recognized by their semiconductor innovations, tasked the team with an intriguing project which involves designing, developing, and assembling a robotic transportation vehicle prototype. Upon completion of the design project, this prototype may be utilized and modified if need be in Texas Instruments’ motor lab located in Dallas, Texas.

The purpose of this project was to develop and configure a functioning prototype vehicle that communicates with the given hardware and software Texas Instruments has provided to the team. The development of this vehicle involves many engineering domains and phases that involve developing and validating communication between a central processor as well as two microcontrollers, and conducting computer programming code written in C. This commands the processor and the microcontrollers with the appropriate movements, incorporating brushless DC motors as well as brushed DC motors which rotate the wheels. Utilizing wireless RF control communication to control the movement of the vehicle and designing a battery powered system to power the vehicle are also included.

Background

Timothy Adcock, the sponsor of the design project, as well as the director of the Texas Instruments’ Motor Lab, has asked for the design and development of a robotic transportation vehicle, which his Motor Lab can use as a prototype. This Motor Lab, located in Texas, is a department where engineers can research new ideas and try to develop these ideas into real-world applications by utilizing many of Texas Instruments’ resources. Many engineers develop their ideas into prototypes here, test and validate them, before potentially creating them into new products. These engineers can make use of the applications, such as motor control algorithms developed in the lab, as well as simulation hardware such as dynamometers, to replicate its performance given different inputs.
Design Specifications and Objectives

Texas Instruments supplied Team 9 with a few of the components needed to complete the design project. One of the items Team 9 had to purchase was the platform of the vehicle which will be assembled, and will configure the components on this platform. The conflicts to be considered are the dimensions of the platform, the weight of the platform, as well as the durability of the platform. The platform dimensions needed to be larger than the microcontrollers, so that the microcontrollers would not be subject to damage. The platform of the vehicle needs to be very durable, so that it will last the length of the semester, and will be able to handle the preliminary tests without any damage.

The movement of the vehicle is controlled by a Chronos watch, which will utilize its RF functionality to communicate with the central processor. This watch will essentially act as a remote control for the vehicle, and based on its motion, will send commands to the central processor with the appropriate movements. The central processor of the vehicle is the Stellaris LaunchPad. The main function of the Stellaris LaunchPad is to interpret the RF signals sent from the Chronos watch, and provide the microcontrollers with the appropriate commands. An interface will be developed between the Stellaris LaunchPad and the two microcontrollers, so that the specific signals/directions received from the Chronos watch will be fluidly and effectively communicated to the microcontrollers. This specific task will be of highest concern and will take majority of the time this semester.

Team 9 will be able to program and interface with the microcontrollers using Code Composer Studio. This software is a Texas Instruments product, which allows the user to develop code and upload it onto the microcontrollers. This software will be used extensively for both the microcontrollers as well as the Stellaris LaunchPad. The program code will feature certain algorithms that will provide the motors with certain RPMs to effectively represent the movement of the Chronos watch. In order to achieve this, the inputs coming from the Stellaris LaunchPad need to be accurately mapped and addressed to variable inputs to the microcontrollers. When constructing this code, Team 9 will also need to consider the speed control of the motors as well as the turn ratios; these will be two important factors that will help determine the success of the project.

Texas Instruments also specified that they would like to implement brushless DC motors, instead of the brushed motors. Texas Instruments also supplied Team 9 with the Graphical User Interface (GUI) and Integrated Development Environment (IDE) for the brushless DC motors named Control Suite. This will be utilized to calibrate the motors and also view the performance. This GUI will allow us to control the output of the DRV8312s to the motors. These motors are very powerful so, certain limits will be established in order to keep the output RPM low.
Chapter 2: Technical Approaches

FAST Diagram

Description of FAST Diagram
The fast diagram above is a visual layout of the project functions for Team 9. The diagram reads from left to right and can be used to get a better understanding of the project. The main function of Team 9 project is to get the robotic vehicle moving. How is moving the frame accomplished? This will be done by moving the frame, rotating the motors, and rotating the wheels. How are rotating the motors and moving the frame accomplished? In order to move the frame it must be supported, as well as its components protected. To rotate the motors Team 9 must control its movements. How is controlling the movement accomplished? Controlling the movement will be done by communication between the central processor and the microcontroller’s. How is the communication from the central processor accomplished? A RF wireless controller as well as autonomous control software allows the central processor to communicate.

House of Quality
Critical Customer Requirements

The house of quality shown above shows the critical customer requirements. The technical importance rating gives a numerical value to what the customer truly is requesting, even though they may not even know. For this project, there were a few major points that were found to be most important to the customer, being the motors, the motor controllers, and the Stellaris LaunchPad.

The first and most important portion of the project is the motors. This functional requirement was spread across almost all of the customer requests. It was strongly related to a working prototype, two wheel drive, reliability, and implementation of brushless motors. With a score of 565.2, it was what had to be focused on the most for the project.

The second and third most important parts were the Stellaris LaunchPad and the DRV8312 EVM. These were found to be slightly behind the motors in value, with scores of 534.8 and 508.7, respectively. The Stellaris LaunchPad was the central controller of the system, and was affected by most of the parts of the project. It was integral in having a working prototype, controlling movement wirelessly, autonomous mode, and reliability. The DRV8312 motor controller was slightly less important but still a
central part of the system. It was most strongly related to a working prototype, two wheel drive, and the implementation of brushless motors.

Some of the least important parts were the DC converters with a score of 173.9, the adaptors with a score of 139.1, and finally the axles with a score of 78.26. These parts, although important when building the prototype, were not important to the customer according to the house of quality. A larger figure can be found in Appendix 3: Figure X.

**Conceptual Design Descriptions**

Team 9 narrowed the decision for a frame to four choices: the Arduino Mobile Platform, the Thumper Mobile Platform, a custom platform from the ECE 480 lab, and a Ford F-150 Custom Cruiser Truck. The Arduino platform is significantly cheaper and fits the desired dimensions needed to mount all of the boards. Also, the frame allows easy access to trouble shoot, and to mount the motors. While these are good qualities, the frame despite fitting Team 9’s desired dimensions, leaves little room for additional components. The frame is also light in weight, which is a disadvantage due to the powerful motors being mounted.

The Thumper Mobile Platform on the contrary, has a bigger frame, which allows easier modification. The platform weighs more, so more of the motor power will be utilized. Also the wheels are bigger allowing the motors to be connected easily. Despite these good qualities there are several setbacks. This frame is significantly more expensive than its counterpart. Also a large portion of what is included in the platform kit will not be utilized, therefore expending the allocated funds inefficiently.

The custom platform from the ECE lab has a bigger frame that both. It is a rectangular metal box, so the components can be placed inside the box and mounted down. It is also sturdy enough to withstand any damage it may take. The customization of this box to Team 9 specific needs is its biggest advantage. But the frames customization is also its disadvantage. Team 9 will have to manually cut holes, axels, place components, secure components, etc. Since this will be a custom design it will take time and effort to get everything running smooth.

The Ford F-150 Custom Cruiser Truck has a smaller frame compared to the custom platform. It is a toy race car that Team 9 can use to hold the components the top can be removed and inside there is enough room to hold everything. A big advantage of the frame is that it was given to Team 9 so it won’t take anything from the budget. Also it comes already assembled so there is little modification that needs to be done on the vehicle. A disadvantage of this frame is that it’s not as durable as the other frames. Most of the truck is plastic so can be damaged.

**EZ430-Chronos Watch**

The robot transportation vehicle will be moved by the radio frequency (RF) control of a Chronos watch. The Chronos watch was made and provided by Texas Instruments. This device can be disassembled and programmed using Code Composer Studio via a USB port. The watch has basic time applications such as,
time, date, alarm, and stopwatch, but also has advanced features that measure: altitude, 3-axis accelerometer, battery voltage, temperature, heart rate, running speed, distanced traveled, and calories burned. The Chronos watch communicates with the Stellaris board, which will be interfaced with the two 8312 microcontroller boards. Since the 8312 microcontrollers regulates the two motors, the RF control of the Chronos watch will allow the user to control the speed and direction of the robot transportation vehicle. Essentially the Chronos watch will be the remote control for the vehicle. The EZ430 Chronos Watch acts as a gyroscope. The Chronos watch will use its built in 3-Axis accelerometer function to track the X,Y, and Z values of the user inputs. This input is defined as the forward, backward, left, and right tilt of the watch. This tilt value can then be used as a variable to transmit to the Stellaris board. So, the user tilt of the watch will dictate the movement of the robot transportation vehicle.

**Code Composer Studio**

The software used to program the Chronos watch, two 8312 microcontrollers, and the Stellaris LaunchPad is Code Composer Studio. This software is an integrated development environment that was created and developed by Texas Instruments for their embedded processor families. The program has applications for development, debugging, compiling, and simulation. This will give the user access to Texas Instruments’ entire device families, essentially any device that has a processor implanted on the printed circuit board will be pre-downloaded on the software. This software is a great tool. It will allow the user to compile, debug, develop, and simulate all Texas Instruments’ devices currently being implemented. Knowing exactly how the printed circuit boards will interact with each other before they are actually programmed will save a great deal of time. Development of a user project will also be simplified due to the fact that all devices can be programmed from common software. There will be no need for compatibility among different programs. One of the biggest advantages this software has is the ability to run situations based on downloaded content. As mentioned the software has access to Texas Instruments’ device families, so the specific device and functionality are available once Code Composer Studio is running. There will be no need for the actually device to be programmed until the functionality of the program code is correct. This will be useful because if there is a crucial error in the user code it can be caught in simulation and not allowed to potentially harm the devices.

**Stellaris LaunchPad**

The central processor of the vehicle will be the Stellaris LaunchPad. The dimensions of this board are 2.0” x 2.25” x 0.425”. This microcontroller was made and provided by Texas Instruments. The Stellaris LaunchPad is a 32-KB ARM microcontroller with 256-KB flash memory, 32-KB SRAM, 80-MHz operation, and USB device. The LaunchPad also has an on board in-circuit debug interface. The Stellaris LaunchPad is a crucial piece to the robot transportation vehicle. This device will receive an incoming digital signal from the RF control of the Chronos watch. After performing a digital to analog conversion it will communicate this analog signal to the 8312 microcontrollers. As mentioned earlier the 8312
microcontrollers will dictate the speed of the motors. So in essence the Stellaris LaunchPad will be responsible for most of the communication in the robot transportation vehicle.

Wheel Choices
There are a variety of wheels that could be implemented in the design. The decision whether to implement omni-directional wheels or standard wheels arose. The omni-wheels allow motion in multiple directions. This will make the steering of the vehicle much easier. However, these wheels are much more expensive than their counterparts (standard wheels). Omni-directional wheels are also harder to mend, and more difficult to mount. In addition, the calculation of the speed could be problematic. Since there are rollers mounted along the wheels’s circumference the torque if reduced. This is important when it comes to getting the vehicle to move from a stand still position. The other option (“standard wheels”) are very cheap and easy to find. The mounting of these wheels is relatively easier and true speed predictable. It has more torque than its counterpart, so it can get the vehicle moving from a resting position. However, the wheels have drawbacks; the wheels only move in one direction and may drag during turning.

Motor Choices
Team 9 had the choice to implement a 24V 4000 RPM Brushless Motors (motor 1), a 24V 185 RPM Geared Brushless Motors (motor 2), or to incorporate a 6V 420 RPM Brushed Motors (motor 3) in the design. Some of the design specifications for motor 1 are; it has 55 watts of rated power, a no load current of 0.4 A, a torque constant of 5 oz-in/A, and weighs 0.94 lbs. The advantage of this motor is it weighs more and is very durable. This is important because Team 9 will need to run extensive test on the motors. A key disadvantage is that the motor way be too powerful. With the motor running at max speed it would move the vehicle too fast to control.

Some of the design specifications of motor 2 are; it has 22 watts of rated power, a no load current of 0.11 A, a stall torque of 55.997 Nm, and weights 0.180 lbs. A key advantage is that the motor is lower in power and has geared motors. This will be much easier to control and its max speed will not move extremely fast. A key disadvantage is that it’s not very durable compared to motor 1. This may become a problem later when Team 9 starts testing. Some of the design specifications of motor 3 are; it has a no load current of 0.80mA, output voltage between three and six, a stall torque of 771 g-cm, and it weights 0.156 lbs. A key advantage to this motor is that it is brushed so it will be significantly easier to control. This is due to the fact the team does not have to worry about 3 phases; all it needs is to be powered. Some disadvantages are that it is less durable than the other motors and it also goes against design specifications.

Ranking of Conceptual Designs

<table>
<thead>
<tr>
<th>Frame</th>
<th>Importance</th>
<th>Arduino Mobile Platform</th>
<th>Thumper Mobile Platform</th>
<th>Custom Design Platform</th>
<th>Ford F-150 Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>
### Budget

The budget was allocated wisely so that there was a substantial amount left over for additional features and potential risks. Texas Instruments provided BLDC motors, BLDC motor drive evaluation boards, and EZ430-Chronos Watch, and a Stellaris LaunchPad. The majority of the allowance went to the wheels, battery, cables and shipping. The wheels and cables were not implemented into the final design. The total cost was $231, which was only 46% of the allowance.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Free (Owned by group member)</td>
</tr>
<tr>
<td>EZ-430-Chronos watch</td>
<td>Free (TI provided)</td>
</tr>
<tr>
<td>Item</td>
<td>Cost</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Stellaris LaunchPad</td>
<td>Free</td>
</tr>
<tr>
<td>DRV8312 BLDC kit</td>
<td>Free</td>
</tr>
<tr>
<td>Wheels</td>
<td>$35.00</td>
</tr>
<tr>
<td>Battery</td>
<td>$30.00</td>
</tr>
<tr>
<td>Cables</td>
<td>$30.00</td>
</tr>
<tr>
<td>Shipping</td>
<td>$86.00</td>
</tr>
<tr>
<td>CC Debugger</td>
<td>$50.00</td>
</tr>
<tr>
<td>Remaining</td>
<td>$269.00</td>
</tr>
</tbody>
</table>

**Budget**

- Platform
- EZ-430-Chronos watch
- Stellaris LaunchPad
- DRV8312 BLDC kit
- Wheels
- Battery
- Cables
- Shipping
- CC Debugger

**Gantt Chart**
Project Management

In preparation for the project undertaking, technical roles were assigned, so that each person could contribute to the overall completion of the project, while highlighting individual strengths of each member. While tasks were assigned to a certain group member, the completion of these specific duties were not be limited to this person, and were shared amongst the group.
The original Gantt chart above displays necessary tasks in order of completion. The schedule was divided up into multiple tasks and time lines in order to find the critical path to completing the project on time. Throughout the course of development, Team 9 deviated from the original path outlined. Due to this and other issues, development of the interfacing between the Chronos watch and the central processor came to a halt and forward progress could not be made although these steps were to be the first tasks to be completed. It was very important to properly interpret the data sent from the watch. The next focus was the use of a graphical user interface to manage and test motor speeds before assembly. This was completed on time and the team was able to discern that the motors having a rated speed of 4000rpm needed to be controlled and reduced via gears, or motor code.

As with most projects, many of the plans were edited and the critical path became delayed. There were several major delays to the project that pushed back the critical path by a large factor. Team 9 foresaw the future risks by referring to the gantt chart and were able to compensate for this beforehand by undertaking tasks that were not scheduled to be completed at that time.

The original frame was vastly too small to be able to carry all of the required contents of the vehicle, so a new frame had to be found, which added unexpected time. The Stellaris connection to the computer did not work on the original board, and time was spent ordering another board. Two days after receiving the second board, it was damaged beyond repair during testing, which required more boards to be shipped, creating an additional time delay.

Possibly the greatest single unexpected delay was the workload for the course outside of the project. There were homework assignments assigned almost every week, which greatly reduced the amount of time early in the course that was given to work on the actual project. For many weeks, more time had to be spent on doing homework assignments that could rather be spent working on the vehicle. The group met three times a week and still found that most of it was going toward other assignments.

The goal for the group was originally to have all of the coding done by Thanksgiving break, and the final two weeks were to be set aside for the remaining assignments including the report, poster and presentation. However, over Thanksgiving break, the team was still waiting for a working Stellaris board to use and changed the way in which the code had to be implemented. This was the critical path, and it put the group behind schedule to complete the project by Design Day.

Chapter 3 - Technical Description of Work Performed

Hardware Design Efforts
As explained previously, most of the electrical system was assigned for use and provided by Texas Instruments. However, design was required to integrate the parts of the system together. In addition, a body had to be created to house all of the components, and mobile power had to be supplied to the electrical system.

The Chronos watch included a wireless communication tool that allowed it to be used as a remote control. This was how the team was directed to control the vehicle. The watch itself was very simple
to use wirelessly. In the presentations the team used the watch as a slide controller, plugging the receiver into the computer with the presentation and using the watch to go to the next slide as the presentation went along.

The receiver plugged into the computer via a USB connection. Naturally the team assumed to plug the USB connector into the USB connection on the Stellaris. The Stellaris, though, had to have ports opened to allow the signal to be sent from the receiver into the Stellaris. If these ports were not opened, all communication would be lost. In order for the Stellaris LaunchPad to decode this data into something it can recognize a CC Debugger had to be obtained. The purpose of a CC Debugger is to interpret flash programming from an IAR system. Team 9 connected the P4 pin of the Stellaris LaunchPad to the receiver to accomplish this. [note appendix]

Other ports had to be opened to connect to the DRV8312 as well. The motor controllers were not too difficult to use, and were quickly figured out using the team computer. The motors were running using a USB connection to the laptop through a control card mounted on the motor controller. The team thought that was how the connection was supposed to be used, but quickly found out that was not the correct way to go about communicating with the Stellaris.

The team delved deep into the datasheets for both the Stellaris LaunchPad and the DRV8312. Communication standards using I/O pins were thought to be the best option. Originally the team was leaning toward using JTAG, but also CAN and SPI interfaces. These were described in great detail in the datasheets, so the team began to look into using them as a means to communicate. These connections however could not be used, because Team 9 could not implement the code in order to utilize the ports.

The schematics on the motor controllers pointed to pulse width modulation, or PWM pins [note appendix]. These seemed to connect directly to the outputs that went to the motors, so they seemed to be a good option. However, after testing, both a Stellaris LaunchPad and a DRV8312 were rendered useless because of a power overload. The PWM connections output 1.6 volts, and the Stellaris, only a 5 volt board, was not expecting that much power. This quickly overloaded the system and forced the team to not only find another solution, but no longer use those boards.

After getting some help, the UART connection turned out to be another option that Team 9 could use for communication. This allowed for a single connection between the two boards. The Stellaris would connect two UART pins to the motor controllers, one going to each controller. This allowed simple communication between the boards. While this allowed quick and simple development of on/off options, it also allowed for a feedback line to be added later for better connectivity. While the hardware layout was mapped out [note appendix], developing the code to control this was an issue Team 9 could not solve in the given amount of time. The vehicle would just be turned on/off as power is given.

So finally Team 9 decided to go back to the PWM pins as stated earlier. This process has the motors running initially, but needed to control the amps going into the boards. Team 9 did this by changing the motors of the vehicle. The new motors ran on 5 volts rather than 24, so a circuit had to be created to convert this [note appendix]. Three pins from the Stellaris LaunchPad were connected to the three PWM pins in the DRV8312 board. [note appendix] These Stellaris connections included P1, P2, P3 ports on the GPIO section of the board.

Another challenge presented in the project was powering the system. Each board had converters to plug into the wall. This was not going to be possible to keep the vehicle mobile, so batteries had to be found for the vehicle.
The team looked into what powered each of the boards. The motor controllers required a 24 volt input, and the Stellaris needed a 5 volt input. This posed a great challenge, as no single power option could do that. The power had to be somehow converted to a different voltage.

The first plan was to find a 24 volt battery to go to each of the motor controllers, and a DC/DC converter would be used to send power to the Stellaris. After searching for the 24 volt battery proved very difficult, the team chose instead to use two 12 volt batteries in series. This would supply the necessary 24 volts, and had several other advantages. Smaller size allowed for more form-fitting in the vehicle. It also gave the team the option to find a DC/DC converter that had an input of either 24 or 12 volts.

Ironically, a DC/DC converter to 5 volts was found, and the input range was 12 to 24 volts. The team had the option to connect the converter to either a single battery to get 12 volts or both to get 24 volts. The team in the end chose to connect it to both for an input of 24 volts because it would allow for similar charge loss of the batteries. If the converter was connected to a single battery, that battery would lose charge faster than the other, making uneven charging cycles.

The design for the frame of the vehicle was left up to the team. Originally a platform from robotshop.com was recommended. The first frame that the team looked into was the Thumper platform. This frame had a big platform plate compared to the others as well as it would be easy to modify because it had precut holes on top of the frame. But the frame was expensive and came with items that were not needed, so the team felt that it was a waste of money.

The next frame Team 9 looked at was the Arduino Mobile Platform. This was a smaller frame and weighed less than the thumper, but it was cheap and didn’t have a lot of extra features that were not needed. Team 9 decided not to go with this frame because of the fact that it took a while to ship and it was smaller than originally thought. Also the frame was not as durable as the team would have liked. This, combined with the fact that the metal plate had to be cut in order for the motors to fit, was the reason why the decision was made to go with another platform.

Next, Team 9 decided to would build a custom platform from a metal box found in the ECE 480 lab. This way it could designed it to specific dimensions as well as have enough room for all the components, since the box was significantly bigger that the previous platforms. The design and layout of this platform was good, but the work that it would take to finish it was becoming an issue due to the semester coming to a close.

Finally Team 9 decided to go with the Ford F-150 Truck platform. A key factor that went into this was that the frame was already assembled and functioning. As it belonged to a team member it was free of charge. The plastic part of the truck can snap on and the components could fit inside without any problems. An added bonus was that the motors that Team 9 would use were already assembled on the frame.

**Hardware Implementation and Photo Documentation**

The given microcontrollers had to be assembled and connected in final form according to the decisions made above. Each of these communication means had to be formed in an efficient format to fit inside of the frame the team chose. Weight distribution was a major concern for the vehicle. Motors had to be mounted on the front to drive the wheels, which gave a large amount of weight to that end of the vehicle. The boards were not very heavy but had to be adjacent to the motors because of the
connections. The counterbalance at the back of the vehicle was the batteries, which were laid out in a way to try to give equal weight to each wheel. The receiver from the Chronos watch also had to be located in a good location so the signal could be received with ease.

The wheels were chosen for specific purposes. The front wheels attached to the motors had good grip, which allowed the motors to get the most out of the torque that was produced. The rear wheels were chosen to be omni-wheels. These had rollers that allowed sideways movement, as the vehicle was to be turned like a tank, spinning the front wheels in opposite direction rather than merely changing the direction of the front wheels.

All of the hardware was secured to the frame, making as robust of a design as possible. This was necessary because the motors ran very high speeds, and a loss of control of the vehicle was possible. Because the frame was designed to fit the needs of the vehicle, the vehicle did not have much wasted space. This made the vehicle maneuverable and also made it look better.

After a group meeting a decision was made to change the frame yet again. The new frame was a Ford F-150 toy car. Based on similar guidelines stated above Team 9 had to make sure that all the components were properly installed as well as powered correctly. In addition, permanent connections needed to be made to all boards to prevent malfunctions during testing. Since this frame came pre-assembled, Team 9 did not have to spend much time altering the frame.

Software Interface, Design and Implementation

The majority of the task specified by Tim Adcock was the interfacing implementation into the design. As a previous vehicle had already been created, additional features were the main point of the project. That included making an ecosystem that included each of the main functions to be included.

Code Composer Studio was used to write programs for the Stellaris LaunchPad. Originally the team tried to download the program from the website, but the version was an older version, and an update was necessary to be able to use it with the Stellaris that was going to be implemented. After contacting the sponsor, a download code was given, and the team downloaded the most recent version of Code Composer Studio onto the team laptop.

The original intent of the team was to develop new code in order to control the Stellaris. It was quickly discovered, however, that this plan would not be feasible. None of the members had much knowledge of how to write code for something like this project, and a different means of coding had to be found.

Some examples were included in Code Composer Studio. Many of these were meant to be tutorials for learning how to use the Stellaris LaunchPad and maximizing its capabilities. Inside of these tutorials were many clues as to how code might be used, but none of them had anything that could be pieced together to develop the code the team needed.

Finally, example files named dev_serial and dev_bulk gave directions for how Team 9 could possible use C code to control their vehicle. The USB dev_bulk example already had code that initializes the UART ports that Team 9 would need based on the hardware connections. (note appendix) A piece of this code is shown below. It shows how to open and update the UART port.

```c
// Open UART0 and show the application name on the UART. //
UARTStdioInit(0);
UARTprintf("033[2JStellaris USB bulk device example\n");
UARTprintf("----------------------------------------\n\n");
```
/ Update the display of bytes transferred. //
UARTprintf("rTx: %d  Rx: %d", ulTxCount, ulRxCount);

After this port was open, Team 9 could write code to open the remaining ports based on (note appendix) this and would allow the proper connection. The problem with this code was that even with the ports open Team 9 was not sure how they would control the speed of the motors, so it would either be on or off.

Team 9 had to come up with another approach to properly control the motors as well as communicate with the Stellaris LaunchPad. The TP pin would either be set high or low. Based on this approach when it is high the receiver and Stellaris communicate, when low there is no communication at all. Also when the watch is tilted the voltage to the motor is increased with a threshold of 5 volts due to the 24V to 5V converter that was built. [not appendix] Below is a part of the code that controls the TP pin.

// This code is to define TP_L and TP High//
void WR1W_WRITE(signed char dat)
{
    signed char i;
    for(i=0;i<8;i++)
    {
        TP_H
        if(dat & 0x01)
            BSP_Delay(1000);
        else
            BSP_Delay(300);
        TP_L;
        BSP_Delay(100);
        dat >>= 1;
    }
}

Problems Encountered

Team 9 began the project trying to account for possible technical difficulties, including power supply issues and ensuring the system was not overloaded, complex code construction, interfacing issues, wireless RF communication stability and errors, and frame damage from loss of control or other crashes. As the project progressed, several other possible risks arose.

A major issue was with the frame choice. The size requirement was underestimated, which lead to the ordering of a frame that was too small for the parts required that had to be towed. The time wasted waiting for that order, along with the time required to find a solution, set Team 9 back. After finding another solution with the custom platform Team 9 thought they had found their frame. But the work needed to complete the frame was also underestimated. Also with Team 9 losing a motor board in testing they decided again to change frames in order to get a working prototype for Design Day.

Another problem Team 9 encountered was complicated C code. Getting the interface between the Stellaris LaunchPad, the Chronos watch, as well as the DRV8312 working correctly was more problematic than Team 9 expected. None of the Team 9 members had experience in working with code composer studio, so the team had to learn a completely new program. C programming can be sensitive to certain inputs. Texas Instruments has special names for particular ports and functions special to their boards. So when problems were encountered in the code the only people that were much help was the TI employees as well as the support community online. This interaction took time and explaining the problem over text turned out to be problematic.
Next the physical connections that had to be made to the boards gave Team 9 problems. Between the all the components there are hundreds of connections, a lot of which could potentially work. But proper implementation of the code would be needed as well. Team 9 had to spend a lot of time looking at diagrams and schematics to get a understanding of what potential connections could be made and where they were. After mapping out potential connections, Team 9 then needed to do testing using a live board. All the boards had to be running at the same time to ensure that everything was running properly. In some cases Team 9 misjudged the voltages and current being spent, and some damage was done to several boards.

Another risk was damage to the boards. Connecting to the Stellaris LaunchPad caused a risk of overheating, melting, or even blowing up the board because of the power being sent through the connection. Also, an unforeseen risk that occurred was a malfunction in connecting with the original Stellaris board. That caused the critical path to be pushed back again because the board had to be shipped, and no coding could be done until the new board arrived. This occurred when connecting to the DRV8312 PWM connections, which output at 1.6 volts, while the Stellaris was only expecting an input in the millivolts range.

All these problems were encountered during this process, but were necessary steps in order for Team 9 to move forward and complete the project. When problems occurred Team 9 handled them together and tried to thoroughly diagnose the problem. In doing so they learned from their mistakes and these failures helped guide them to a better solution. Unfortunately with this knowledge came sacrifice. Team 9 spent a great deal of time tackling the problems they encountered, which threw them off their Gantt chart critical path.

Chapter 4: Engineering Data and Results

Data and Results

Using the equation $E_{out} = D \cdot E_{in}$, the duty cycle to convert the 24 V source to 6V was calculated to be .25.
When E\_in=24\,\text{V}, \, D=0.25, \, E\_out \, i s \, 6\,\text{V}. Calculations to convert different input voltage values were calculated and each voltage measured using a Digital Multimeter. The results of this test are shown below. The down voltage conversion was required to power the motors and the Stellaris LaunchPad without damaging them. The circuits to implement the DC to DC converter were constructed based off of the of the schematics to the left. Another useful tool was PSpice which was used to calculate circuit outputs before assembling and measuring values. This enabled us to troubleshoot circuit construction without the hassle of wondering if the designed circuit was correct. This helped to save time.

![DC-DC Converter Circuit](image)

Figure X: DC - DC Converter Circuit
Phase A of DRV 8312 is used as the IGBT switch of the DC-DC converter. This output of DRV8312 (phase A) is controlled by PWM signal with 0.25 Duty Cycle @10kHz.
This waveform is the measured output of the circuit after the inductor was added. The inductor is used to decrease the ripple current.

This waveform is the measured output of the circuit after the capacitor was connected. The capacitor is used to reduce the voltage ripple to get a more stable DC output. This reading or measurement was done using an Aglient Infinium Oscilloscope.
This is a screen shot of the signals sent to the PWModulators of the DRV8312 as ran by the times.c Code (Appendix 3: ). It demonstrates the use of the timers to generate periodic interrupts. One timer is set up to interrupt once per second and the other to interrupt twice per second; each interrupt handler will toggle its own indicator on the display. UART0, connected to the Stellaris Virtual Serial Port and running at 115,200, 8-N-1, is used to display messages.
Figure X. Square Wave signal with .25 Duty Cycle for DC to DC conversion from 24V to 5V.
Figure X. Square Wave Signal with Duty Cycle of .15 for DC to DC Voltage Conversion.
Final Prototype Description

After much time of experimenting and tinkering to find out what worked and what did not work, the team in the end settled on a final design for the prototype. The required parts of the system, including the Chronos watch input, the Stellaris LaunchPad central controller, and the DRV8312 motor controller were used. This was all assembled inside of the Ford F-150 frame, which had been modified and painted by the team. Connections were made between each part in an organized manner so that when the vehicle was opened it was easy to identify how the connections between the parts were made. Also put into the vehicle were two batteries and the DC-DC converter, which supplied the base power to the vehicle while remaining in a rearward location to offset the large weight of the motors in the front.

The team finally decided to use different motors than were specified in the report. Although the team had gotten the brushless DC motors working, damage to the controller board would not allow a functional prototype to be displayed for the deadline. As previously explained, the team decided that a working prototype was more important in this situation, so alternate motors were used. They were not brushless, and allowed for forward, left, and right movement.

All of the code was flashed onto the Stellaris from the laptop for the Stellaris to function as the central controller. Additionally, the CC Debugger was placed on the vehicle to allow the communication between the Chronos watch and the Stellaris to be understood. Finally, after much testing, the prototype was complete and ready for the demonstration on Design Day.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>12V</td>
<td>6.0789V</td>
</tr>
<tr>
<td>13V</td>
<td>6.0817V</td>
</tr>
<tr>
<td>14V</td>
<td>6.0849V</td>
</tr>
<tr>
<td>15V</td>
<td>6.0882V</td>
</tr>
<tr>
<td>16V</td>
<td>6.0920V</td>
</tr>
<tr>
<td>17V</td>
<td>6.0956V</td>
</tr>
<tr>
<td>18V</td>
<td>6.0998V</td>
</tr>
<tr>
<td>19V</td>
<td>6.0044V</td>
</tr>
<tr>
<td>20V</td>
<td>6.0082V</td>
</tr>
<tr>
<td>21V</td>
<td>6.0126V</td>
</tr>
<tr>
<td>22V</td>
<td>6.0159V</td>
</tr>
<tr>
<td>23V</td>
<td>6.0172V</td>
</tr>
<tr>
<td>24V</td>
<td>6.0174V</td>
</tr>
</tbody>
</table>

Figure X. Testing Result of DC Conversion Circuit

These are voltage readings from the DC to DC conversion during testing. The test was conducted using an Agilent Digital Multimeter. The table is a conversion from 12V to 6V all the way to 24V to 6V.
Chapter 5: Conclusion

Conclusion

Although the majority of the design project was a great success, there were many challenges and difficulties that the group faced throughout the semester. Towards the end of the semester, a few of the components were damaged during testing causing great concern. Due to the damage of the equipment and the lack of time, there were a few adjustments the team decided to make in order to have a working prototype for design day. This included the inability to move in the backwards direction, as well as the incorporation of brushed DC motors instead of the brushless DC motors. This was a major design decision the team faced.

The group however was able to create a working prototype for Texas Instruments despite the obstacles the team faced. They were able to wirelessly communicate with the Stellaris LaunchPad, and developed effective programming code, which rotated the wheels according to the watch’s tilt. This accomplishment along with the construction of the platform and the power distribution among electrical components were significant achievements. They were also able to successfully manage a budget and deliver the product to the sponsor in time.

The design team would recommend future students similar such projects for them to develop their technical knowledge. Robotics development is an area that is rapidly growing, and such projects provide students with the necessary expertise to participate in larger programs. Texas Instruments is also a great company to work with and provide design teams with the technical support they need.

Overall, design team nine completed all of the necessary objectives and gained much experience in teamwork, design, project and budget management, and delivering a product before a deadline. These are all learning experiences the group members plan on utilizing as they grow and mature in their perspective engineering fields.
Appendix 1 - Technical Roles & Responsibilities
Alex Koschmann - Team Manager

Alex’s contribution was very dynamic as a member of team 9. The group collaborated on most of the project, highlighting how well each member worked inside of the group. Alex took both leadership roles and assisting roles, filling in wherever necessary to work on the project throughout the semester.

One of Alex’s main contributions was the development of the movement design for the vehicle. Culminating in the movement diagram in the appendix, Alex took the requirements of the movement in the project and thought out how to use them mathematically within the program. He thought through how the Chronos watch gyroscope would work and how the motor would take inputs, and finally figured a way to get those two communication lines synchronized in a clean fashion. He thought about how a user would try to control the vehicle and how the vehicle should react, and developed a method to ensure that the movement would work in such a manner.

Alex also contributed in testing and assembling the vehicle, helping debug both the Stellaris LaunchPad and the DRV8312 motor controllers. There were many problems with the boards, and Alex, along with the other members, slowly but methodically tested the problems to discover root causes and how to fix them. Alex also assisted assembling hardware of the vehicle, making wired connections, soldering parts, and designing the layout of each component.

Throughout the semester, Alex planned several conference call meetings with the team sponsor, Tim Adcock, from Texas Instruments. Alex was able to effectively manage schedule planning, allowing the team to meet regularly so that all of the work that was needed had enough time to be completed.

Alex did in-depth research about the Stellaris LaunchPad with regards to connecting the board to the DRV8312 motor controller. He read all of the datasheets and guides for both boards, and learned means of connecting the parts together so they could communicate effectively. Alex also learned about different means of communication such as SPI, I2C, JTAG, and CAN, and helped the team decide which method would be best for communication.
Haotian Cai – Webmaster

Haotian is very important to the ECE 480 design 9. His non-technological role this semester was Webmaster. He managed the team website and kept updating the progress of the design. The more important contributions to the design team 9 was his work with the Chronos watch. Haotian gave important information to the team on how to design the robot. He suggested building and assembling the robot by using individual parts instead of buying the whole frame. The first reason was the team could spend less money from the $500 budget. Another reason was it was easier to mount the motor by drilling the holes. The team could build the robot as they wanted it to be. He also helped solve the power issues for the team. The Launch Pad was powered by 5V DC. However, the BLDC motor worked under 24V DC. Haotian found the DC-DC converter to convert 24V to 5V DC power. He used the IGBT gate controlled by the PWM signal generated by Launch Pad to convert the 24V DC to 6V DC in order to power the DC motor in the final design. He built a LC filter to make 6V DC more stable (Buck DC-DC converter).

For the coding portion, Haotian figured out physical connections for communication between receiver and the Launch Pad. Also he participated in the main code programming. With the hard work of the whole team, Team 9 built the interface of the Launchpad, receiver to control the motor. He developed the remote control function by using the acceleration sensor built in the watch. The final design allowed the robot to go forward and backward, left and right controlled by moving the watch to different angles.

In addition, Haotian did a lot of work on assembling and testing. He placed parts into the robot and made the wire connection with each part. All parts were tested several times to make sure they were working well by Haotian. He showed the test data to the team and helped other team members with their work.
As a member of Team 9 Stevie contributed in a variety of different ways to ensure that the project was completed. The project was very interconnected so the team spent a lot of time collaborating together and exchanging ideas. Stevie was involved in the design process of the frame, as well as the hardware/software integration. Stevie spend a large part of his time learning code composer studio and working on the code to develop a working interface for the transportation vehicle. With this included making sure all the proper software was downloaded so that programs ran correctly. Stevie learned how to navigate code composer studio, learning its functions and applications. In doing so he was able to figure out how to successfully open a new project with the correct parameters, as well as import example products onto code composer studio. This allowed Stevie to successfully compile working C+ code on to the Stellaris LaunchPad. Writing the code was a critical part of the project and the communication between all the boards. Stevie looked through sample code in the StellarisWare folder to try and implement some of this code into his project. This example code came with the download of code composer studio. He then took an in depth look into the USB_dev_bulk example and USB_dev_serial example. With these examples he was able to pin point the code the opened the UART ports that were going to be used to establish communications the Stellaris LaunchPad, DRV8312, and the Chronos Watch. Stevie along with other members successfully figured out how to power our boards as well as how much power was needed to run all the boards correctly. With this testing needing to be done as well as a understanding of the diagrams and schematics. Stevie played a part in both of those processes working with other group members to ensure that progress was being made. Finally Stevie played a big part in researching different platforms that team 9 needed to consider for their frame. With this he helped establish pros and cons of each frame to better determine which one was the better choice. In all, the project was a very interconnected problem in which the team worked collaboratively to figure out. Stevie was a part of this collaboration to help his team solve their project assigned.
Timur played an integral role as a member of design team nine and contributed to many different facets of the project. First and foremost, acting as the presentation preparer, Timur created stimulating powerpoint presentations for the design team to present. He also played a major role in leading conference call discussions with the sponsor and helped create a working relationship with Texas Instruments. By communicating effectively, he was able to maintain close contact with the sponsor, and periodically informed him with relevant updates. He also remained in close contact with the Electrical and Computer Engineering Department, in order to reserve rooms for group meetings and conference calls. Also within the department, he continuously kept in contact with the purchasing order manager, in order to deliver team goods on time. Outside of the communication functions he handled, he took control of the hardware connections between electrical components that the project required. After thorough testing and consulting with team members, he proposed a constructive solution for the team. In addition, he assisted in the assembly of the platform of the robotic transportation vehicle as well as purchasing certain components the group needed. Timur also provided the team with solutions regarding weight and power distribution of the vehicle. During the last week of the course, Timur constructed an intriguing poster for which the design team will present during Design Day and the department will use for future reference. Overall, Timur was a vital team member of the group by performing various tasks, all of which were beneficial for the group’s success.
Grace Jones was the lab coordinator for ECE480 Design Team 9. Her role included but was not limited to keeping the lab equipment organized, finding, organizing and assembling parts, and overseeing safe integration with power systems. Grace’s contribution to the group was important. She coordinated and kept accurate documentation of team meetings in order to keep the team on track and also constructed team document templates for improved aesthetic appeal. In addition, Grace maintained an informal log book of up to date designs to support group progress and discussions. This was done to prevent the team from being stagnant, while dealing with small issues previously discussed, and decisions that were already made.

Research into all components and their functionality was an enormous feat. Grace as well as other members of Design Team 9, worked to understand schematics and data sheets of all components in order to fully understand the task at hand as well as to explain findings and reasonings to group members in efforts to keep all group members up to speed on technical issues.

Grace worked on each portion of the project including EZ-430 Chronos watch integration, power system design, prototype frame and final frame construction, Stellaris Integration, and DRV8312 integration. All of these measures required extensive research and sub tasks such as soldering simple circuits, and making proper connections between the different elements. Grace soldered many connections, and made power adapters for each 8312 and help to develop functioning code. She also helped to design and construct the frame, debug and write startup code to the Stellaris LaunchPad and helped to figure out a powering scheme for the Stellaris that did not include connecting the device to the computer.

Grace and the team worked together to create a working prototype. This required long nights spent in the lab, and a lot of teamwork and cohesion. Grace sought to foster this sense of cohesion, by recommending team building exercises such as spending time outside of the lab at a team dinner in order to discuss issues and ideas due to the many frustrating unexpected occurrences during the course of the project.
Appendix 2 - References
References

Appendix 3 - Technical Attachments
Figure 1: Flow Chart of Team 9 Transportation Robotic Vehicle. This figure gives a pictorial layout of the project.
Interfacing

Program/Algorithm Design

The interfacing portion of the project will be completed in Code Composer Studio using the C programming language. An algorithm has been developed by Team 9 to transfer the information received from the chronos watch to the microcontrollers using a central controller. The first part of the C program takes the x and y values as inputs from the chronos watch. These values can be stored as simple variables. From there the direction is found. A chart has been developed to display how the decision will be made. First, a 0 section for no movement is at the center. This is bordered by a threshold of x and y values. Until those values of x or y are crossed, the vehicle will not move. However, if those values are exceeded, the direction is then found, being one of the following: forward, backward, left, or right.

After the direction has been found, the next major part of the algorithm is to find the RPM that needs to be output to the motors, which will be sent to output ports, given in the algorithm as variables a and b. To solve for the RPM values, equations will be developed for each direction. If the direction is found to be 0, no equation will be needed, as the output value will be 0 to each motor. For the RPM equations, several important notes must be mentioned. After the 0 threshold is crossed, the RPM value will increase linearly, beginning at 0. The motors have an extremely high maximum RPM, so the equations must be configured so that the RPM is limited to a safer and more controlled value. The equation will need to have this maximum value built in so that a safe value cannot be exceeded. Once the equation reaches the RPM limit, the motor will continue at that rotational speed, no matter how much further the watch is turned.

As the final part of the program, the values will be sent to the ports, which will then be sent to the through microcontrollers to the motors. These values will be dynamically changed as the inputs x and y change, so real time adjustments will be made to the motors, which will allow fully functional remote control of the vehicle.
Figure 2A: 24V 185 RPM motor. This motor was considered for Team 9 project.

Figure 2B: 6V 425 RPM motor. This motor is used in Team 9 final prototype.

Figure 2C: 24V 4000RPM motor. This motor was used in frame three of Team 9.

Figure 3A: Layout and dimensions of the 6V brushed motor.
Figure 3B: Layout and dimensions of the 24V 4000 RPM brushless motor.

Figure 3C: Layout and dimensions of 24V 185 RPM brushless motor.
Figure 4A: Thumper Mobile Platform. Initial frame for Team 9 project.

Figure 4B: Arduino Mobile Platform. Second frame for Team 9 project.

Figure 4C: Custom Platform. Third frame for Team 9 project.

Figure 4D: F-150 Custom Cruiser Truck. Final frame for Team 9 project.
Figure 5: DRV8312 Schematic. This schematic shows vital connections that lead to the motor inputs. Team 9 used this to help map out the connections that were made for their vehicle.
Figure 6: 3 Phase Motor Diagrams. This diagram shows the inner components of the 24V 4000 RPM motor. Phase A of the DRV8312 is used as the IGBT switch for the converter circuit. This output (Phase A) is controlled by PWM signal with 0.25 Duty Cycle at 10KHz.
Figure 7: DRV8312 Board Layout. This figure shows the important regions of the 8312 board as well as their names used.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board supply voltage</td>
<td>4.75–5.25 V\textsubscript{DC} from one of the following sources:  \begin{itemize}  \item Debugger (ICDI) USB Micro-B cable (connected to a PC)  \item USB Device Micro-B cable (connected to a PC) \end{itemize}</td>
</tr>
<tr>
<td>Dimensions</td>
<td>2.0” x 2.25” x 0.425” (L x W x H)</td>
</tr>
<tr>
<td>Break-out power output</td>
<td>\begin{itemize}  \item 3.3 V\textsubscript{DC} (300 mA max) \item 5.0 V\textsubscript{DC} (depends on 3.3 V\textsubscript{DC} usage, 23 mA - 323 mA) \end{itemize}</td>
</tr>
<tr>
<td>RoHS status</td>
<td>Compliant</td>
</tr>
</tbody>
</table>

Figure 8: Stellaris Specification Table. This figure shows general information about the Stellaris LaunchPad. Team 9 utilizes some of this information, such as the board power supply to safely run tests on prototype.
Figure 9: Stellaris LaunchPad Flowchart. This figure shows a basic flowchart of how the Stellaris components interact with each other. Team 9 used this to help write software for the project.
Figure 10: Stellaris LaunchPad Schematic. This figure shows the USB connection of the Stellaris. Team 9 used this schematic to track the USB connections to see what potential ports could be available.

### Table 2-1. USB Device Signals

<table>
<thead>
<tr>
<th>GPIO Pin</th>
<th>Pin Function</th>
<th>USB Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD4</td>
<td>USB0DM</td>
<td>D-</td>
</tr>
<tr>
<td>PD5</td>
<td>USB0DP</td>
<td>D+</td>
</tr>
</tbody>
</table>

Figure 11: USB Table. This table shows the D+ and D- pin names. These pins are because they are used to send the data across a USB port.
Figure 12: RF Communication Snapshot. This figure is a snapshot of the signal produces as the Chronos watch is tilted.

Figure 2-5. eZ430-Chronos Control Center With Acceleration Data
3.6.3.4.1 RF Access Point Update

The CC Debugger is required to re-program the CC1111 based RF access point. To perform the update, the SmartRF Flash Programmer software is used to on the PC side.

1. Solder a debug interface connector cable onto the CC1111 access point as shown in Figure 3-15.

<table>
<thead>
<tr>
<th>CC-Debugger</th>
<th>RF Access Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin 1</td>
<td>GND</td>
</tr>
<tr>
<td>Pin 2</td>
<td>Vcc</td>
</tr>
<tr>
<td>Pin 3</td>
<td>P2.2 (DC)</td>
</tr>
<tr>
<td>Pin 4</td>
<td>P2.1 (DD)</td>
</tr>
<tr>
<td>Pin 7</td>
<td>RST</td>
</tr>
<tr>
<td>Pin 9</td>
<td>Vcc</td>
</tr>
</tbody>
</table>

Figure 3-15: eZ430-Chronos RF Access Point Connection to CC Debugger

Figure 13: RF Communication. This is a layout of the RF USB on the Chronos watch. Team 9 used this to establish a means for data transmission from the watch which is further translated into a format that the Stellaris LaunchPad could understand. This figure also shows the pin layout as well.
Figure 14: UART Hardware Connections. This is a layout of how Team 9 planned on using the UART ports to establish a hardware connection between the Chronos watch, Stellaris, and DR8312 board.
Establishing Communication Between the Watch and the Receiver

The receiver will get a lot of data from the watch. So the code will filter the X, Y value of the acceleration sensor sent by the watch. The data from the X, Y value is 8 bit. The first digits are used to detect the movement of the watch. When the X, Y values are filtered by the receiver, the data will be sent by TP pin. TP_H is defined as 1, and TP_L is defined as 0 to send the binary data to the Launchpad. When the Launchpad gets the binary data from the receiver, the codes will analyze the data. If the X is positive, the Launchpad will generate a PWM signal with duty cycle between 0.15-0.25 to pin PF1. This is to control the robot moving forward. If the Y value is positive or negative, the Launch pad will generate two PWM signal to control the half bridge IGBT switch as well as reduce the voltage from 24V to 5V.

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    // This code to make sure the receiver will connect with the watch whenever the watch on.
    if (!simpliciti_on)
    {
        system_status = HW_SIMPLICITI_TRYING_TO_LINK;
        simpliciti_start_rx_only_now = 1;
        simpliciti_start_now = 1;
    }

    // This code is to define TP_L and TP High
    void WR1W_WRITE(signed char dat)
    {
        signed char i;
        for(i=0; i<8; i++)
        {
            TP_H;
            if(dat & 0x01)
                BSP_Delay(1000);
            else
                BSP_Delay(300);
            TP_L;
            BSP_Delay(100);
            dat >>= 1;
        }
    }
```
//BSP_Delay(20);
}

//Part of Code for Launch Pad
signed short WRW_READ()
{
    unsigned char i;
    unsigned short dat;
    tr_begin = 1;
    while(tr_begin);
    for(i=0;i<8;i++)
    {
        change = 1;
        while(change);
        if(wrw_err)
        {
            SysCtlDelay(ROM_SysCtlClockGet()/1000);
            wrw_err = 0;
            return 0x00;
        }
        dat >>= 1;
        if(rcvbit)
        {
            dat |= 0x80;
        }
        else
        {
            dat |= 0x00;
        }
    }
    tr_begin = 1;
    return(dat);
}

void Timer0IntHandler(void)
{
    ROM_TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    ROM_IntMasterDisable();
    curr_sample = !!(GPIO_PinRead(GPIO_PORTF_BASE, GPIO_PIN_4));
if((curr_sample == 0)&&(last_sample == 1))  //
{
    if((counter > 40) && (counter <= 80))  //Bit "1"
    {
        rcvbit = 1;
        change=0;
        wrw_err = 0;
    }
    else if((counter > 0) && (counter <= 40))  //Bit "0"
    {
        rcvbit = 0;
        change=0;
        wrw_err = 0;
    }
    else if((counter > 80) && (counter < 150))  //Start of byte
    {
        tr_begin=0;
        wrw_err = 0;
    }
    else
    {
        wrw_err = 1;
    }
}
else if((curr_sample == 1)&&(last_sample == 0))  //
{
    counter = 0;
}
else
{
    if(counter > 150)
    {
        counter = 0;
        wrw_err = 1;
    }
}
counter++;

last_sample = curr_sample;
ROM_IntMasterEnable();

// To convert the data and tell is positive or negative and define the
moving directions fb, lr.
rcvdata = WRW_READ();
lr_rcvdata = (rcvdata & 0xf0);
fb_rcvdata = ((rcvdata & 0x0f) << 4);

if(fb_rcvdata < -10)
{
    HWREGBITW(&g_ulFlags, 3) = b_light;
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, g_ulFlags);
    lighton_delay(fb_spd);
}
if(lr_rcvdata > 10)
{
    HWREGBITW(&g_ulFlags, 2) = r_light;
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, g_ulFlags);
    SysCtlDelay(ROM_SysCtlClockGet()/1000000);
}
else if (lr_rcvdata < -10)
{
    HWREGBITW(&g_ulFlags, 1) = b_light;
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, g_ulFlags);
    SysCtlDelay(ROM_SysCtlClockGet()/1000000);
}
else
{
    HWREGBITW(&g_ulFlags, 1) = ~r_light;
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, g_ulFlags);
    HWREGBITW(&g_ulFlags, 2) = ~b_light;
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, g_ulFlags);
    SysCtlDelay(ROM_SysCtlClockGet()/1000000);
}
}