Aaron Zamora

Brandon Champagne

Don Nguyen

Cullen College of Engineering

4800 Calhoun Road

Houston, Texas 77002

December 8, 2015

Dr. John Glover

Cullen College of Engineering

4800 Calhoun Road

Houston, Texas 77002

Final Report for Sparkfun Autonomous Vehicle Competition:

Dear Dr. John Glover,

This final bi-annual report outlines our team’s current completed milestones. The contents of the report include our current budget along with a visual aid showing the completed milestones, an overview of the current design and plans moving forward.

Regarding the overall progress of this project, some of the milestones we have accomplished are communication between the pixy and Tiva-c, as well as differentiation between red barrels and red stanchions. We are ready to demo the robot at your earliest convenience. Please contact us, if you have any questions or concerns.

Best,

Aaron Zamora

|  |  |
| --- | --- |
| https://sparkfun.com/marcomm/SF-Flame-1C-01.jpg  Sparkfun AVc 2016 |  |

Sparkfun AVC 2016

Team Members: Brandon Champagne

Don Nguyen

Aaron Zamora

Project Sponsor: Dr. John Glover

**Abstract**

This document serves as a progress report for Team 9’s project with the Sparkfun Autonomous Vehicle Competition. The purpose of the project is to bring recognition to the Cullen College of Engineering, by demonstrating safe autonomous driving in the contest. The preliminary design includes two Tiva-C microcontrollers, 4 distance sensors, a PIXY camera, and an accelerometer. Input to the system will be provided via infrared sensors located on the front and sides of the vehicle, and height, width, & color collected through the Pixy camera.

The competition points are awarded by interacting with objects along the course, and completing the course in under 5 minutes without the use of GPS. Additional points are awarded for each second a team is under time. The obstacles on the track area are a green hoop, blue ramp, yellow & red stanchion. Each object has a task associated with navigation of the track. The Pixy is used to identify objects’ locations; one of the Tiva-C’s will be used to control the direction in which the vehicle is moving. The other Tiva-C is controlling the speed at which the vehicle is traveling, ensuring it does not collide with objects.

Communication between the Tiva-C and PIXY is accomplished using I2C. For the testing of the infrared sensor we performed stopping distance trails. Concluding the distance trials, the average distance was calculate as 22.4[ft]. This will not suffice for the barrel obstacles which are 10[ft] apart. This can cause problems when the vehicle is weaving in and out of the barrels. Testing of the accelerator, steering and trim settings was performed to check if the measured PWM matched the values on the data sheet. The datasheet specifies a frequency of 1600-1700 [Hz] and a period of .588-.625[ms]. Our measured values have a period of 9.87[ms] which contradict the data sheet, moving forward we will use the measured values to set values via software, for the speed and direction of vehicle. Current accomplishments includes both microprocessors operating the vehicles basic functions, direction, motor controls, recognition of red barrels and red stanchion, and communication between the devices. For the completion of the vehicle we need to recognize yellow stanchion, green hoops and blue ramps.

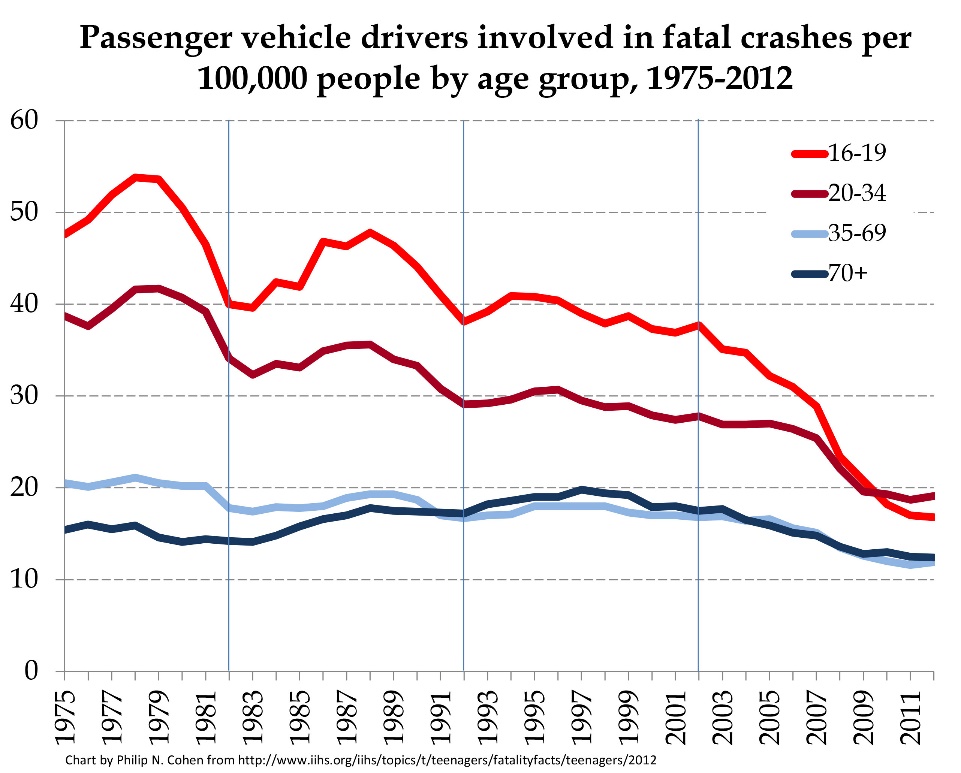
ii

# **Purpose**

This project is based on the Sparkfun Autonomous Vehicle Competition. The purpose is to bring recognition to the Cullen College of Engineering, by demonstrating safe autonomous driving in the contest. To compete, teams must build a vehicle that will interact with obstacles on the course to accumulate points, and finish the race. Points are awarded for completing the course in under 5 minutes (1 point per second), passing through a hoop (50 points), clearing a ramp (50 points), using a shortcut (150 points), not using GPS(150 points), and clearing corners (25 points per corner). [1] To accomplish this the team has designed a system to be implemented with an RC car.

Problem

The number of vehicular crashes worldwide is a staggering 1.3 million road crash deaths per year, being the leading cause of death among ages 15-29. The United States figures are 37,000 road death per year and cost our government $230.6 billion per year. Figure 1 shows the history of these statistics.



1

*Figure 1- Number of fatal vehicular deaths*

Around 2008 you can see a noticeable dip in numbers. This is due to the implementation of assisted driving. These systems aim to reduce human error by means of alerts such as head on collision warnings, as well as vision systems such as side view cameras. Autonomous vehicles can reduce these staggering numbers with the elimination of human error altogether, thus making transportation safer.

*Recognition:*

The Sparkfun AVC is an annual event that brings robotics teams together at Sparkfun’s headquarters. Winning the competition will bring the Electrical and Computer Engineering (ECE) UH engineering program national recognition, When UH engineering students enter competitions such as Sparkfun and win, it shows the strength of our ECE engineering program.

**User Analysis**

Autonomous controls will be used by car manufacturing companies to design better full sized driverless vehicles. Driverless vehicles in the future will provide a traffic free, safe and efficient form of transportation. The only necessary qualifications for the use of a driverless vehicle will be knowing how to enter destinations. As for the Sparkfun AVC, to use the vehicle a user will need to know how to press a button which starts the race. To be able to develop the vehicle further, one should have a background in C programing and be familiar with the Tiva-C and Pixy camera.

# **Overview**

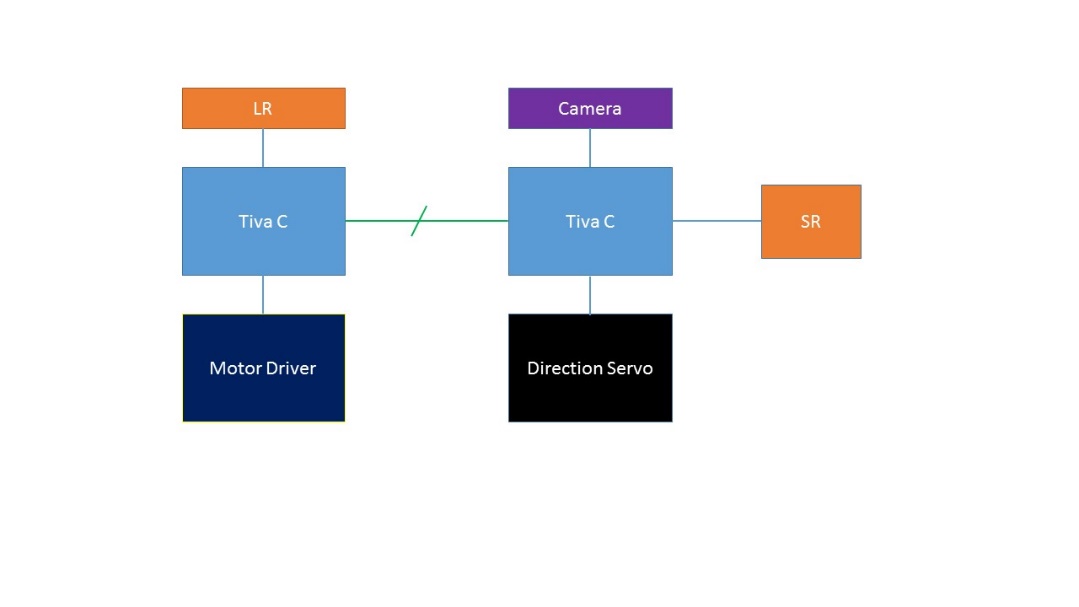
The overall goal for this project is to autonomously navigate the track that is given to us in the competition. We have modeled our design off of last year’s competition. This year’s rules will be released in February 2016. In order for the vehicle to autonomously navigate the track it needs to recognize and avoid objects. Figure 2 shows the vehicle’s POV as it navigates the track while interacting with obstacles.

2



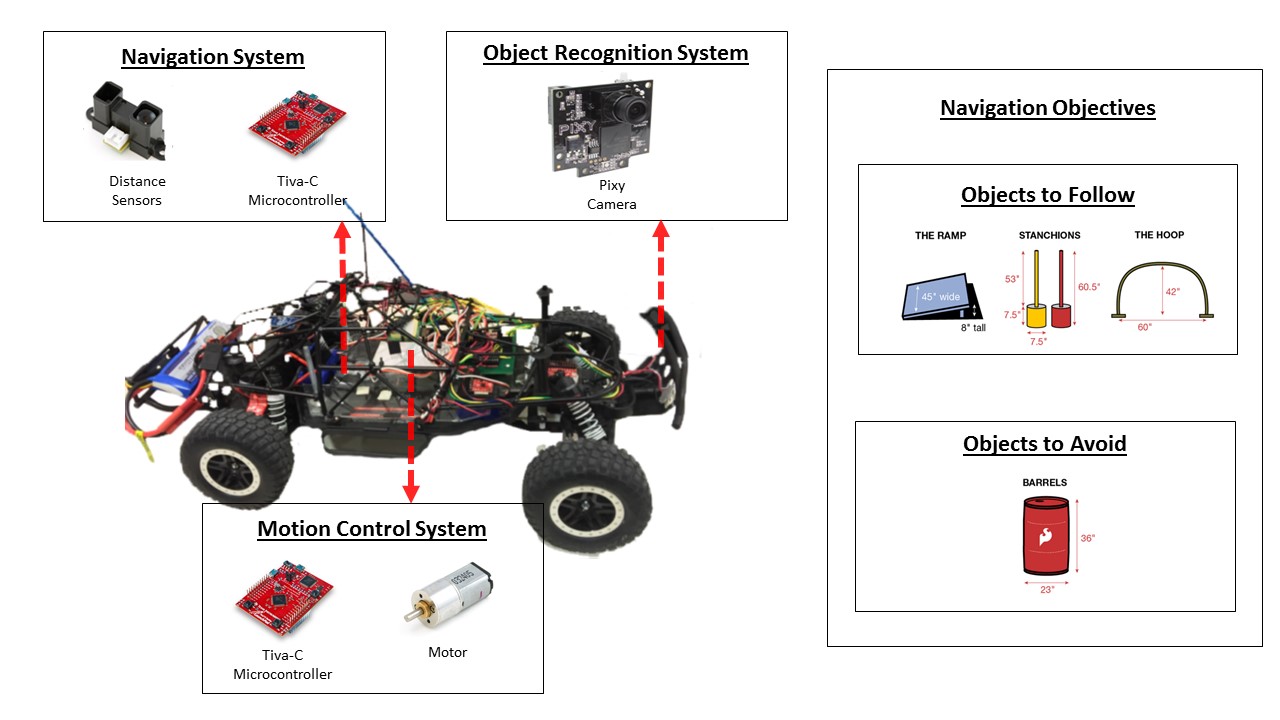
*Figure 2- Robot POV of Sparkfun AVC 2015*

For the design, driving is simplified into two processes much like driving a full sized vehicle, controlling forward speed through accelerating and braking, and steering. For the execution of the two tasks we decided to delegate the activities to two processors. One processor continuously looks in front of the vehicle to control the speed to avoid running into an object, and the other processor controlling the direction. Communication between the two Tiva-c and the camera is important to handle the accelerometer data as well as the object data. Figure 6 shows a block diagram representation of the communication between the devices.



***Figure 3:*** *Design of Communication between hardware and software. One* *Tiva-C will take a sample, calculate forward speed and drive motor while the second Tiva–C will get the picture data from the PIXY camera and calculate optimal direction.*

3

To accomplish vehicular control, long range distance sensors are used to acquire data to calculate a safe forward speed. The PIXY camera aids in not only directional navigation, but also identifying objects that require different actions. Short range distance sensors are used on the sides of the vehicle to detect objects that are not in view of the camera. The design for our robot is shown in Figure 4 below. 

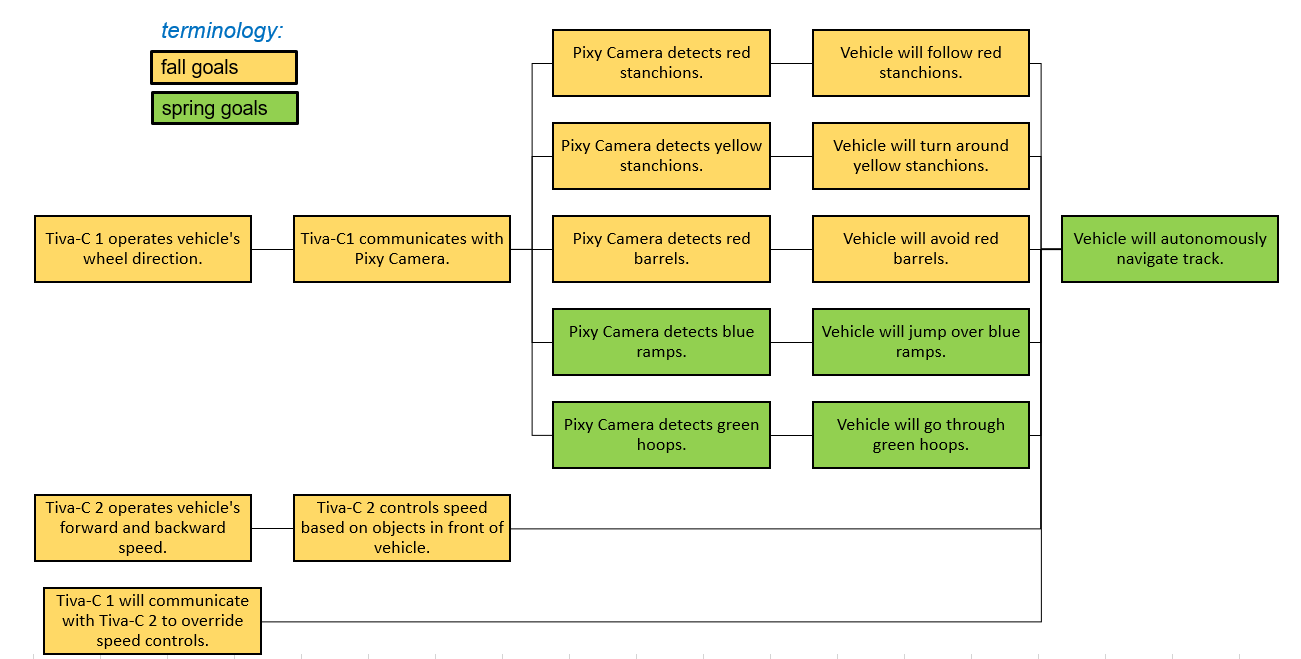
**Figure 4:** Overview Diagram depicting the basic design for the autonomous vehicle.

Object recognition in the competition is vital to acquire points, since each obstacle has a different set of instructions to follow. The green hoop obstacle is the only object on the track that is required to go through to accumulate points. The blue ramp is an obstacle that the vehicle must locate and drive over. The yellow stanchions are located on every corner while the red stanchions form the perimeter of the track. The red barrels are to be weaved in and out of, if they are hit or knocked over, points will be deducted. Communicating between the two microprocessors, is required for the vehicle to differentiate the different commands for a certain object. Tiva-C was chosen for processing because of its real time operating system, which would allow the vehicle to react faster to the environment around it. The communication protocol between the devices is I2C.

4

# **Target Objective**

Goal Analysis for the fall semester is located below in Figure 8 the overall target objective is to have an autonomous vehicle to navigate the track. The fall goals are in yellow and Spring goals are in green. For the fall our goals are to have a vehicle to recognize stanchions, and red barrels by communicating between the Tiva-C and PIXY.



**Figure 5**: Goal Analysis for Sparkfun AVC 2016 project

For the Fall goals we have completed all the goals indicated by the strike through them in Figure 8 in the statement of accomplishments. In the final week leading up to the demo for the fall we are working on getting the object recognition to 80% accuracy. Beginning in Spring semester, we will begin to right algorithms to detect green hoops and blue ramps in hopes that it can complete a track by march of next year.

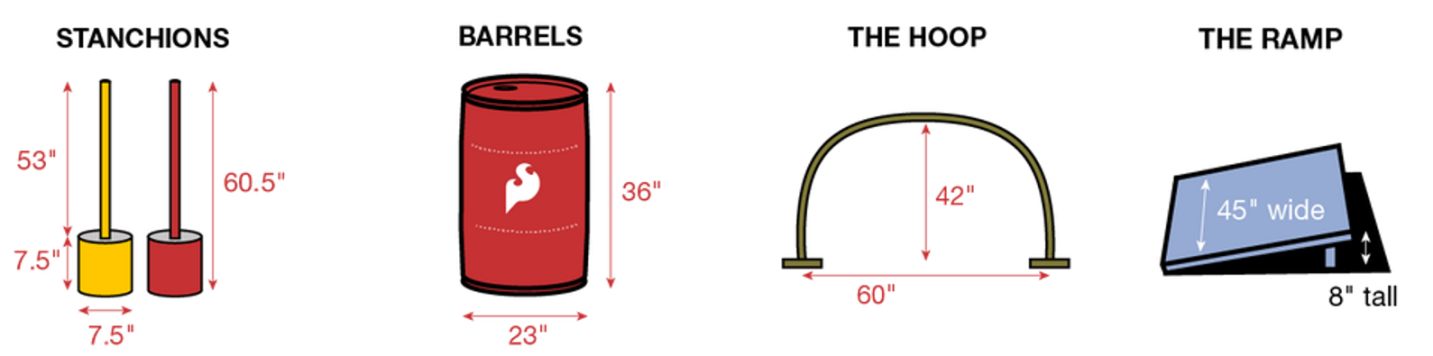
# **Specifications**

Specifications were given for the Sparkfun AVC. The specifications given were the dimensions and layout of the track, depicted in Figure 3, and the dimensions and colors of the obstacles shown in Figure 4. The minimum speed is 2.98 [ft/s] to complete the track in under 5 minutes. Different speed consideration and point accumulation is shown in Table 3 below.

5



**Figure 6:** Sparkfun AVC 2015 track [2]



**Figure 7**: Sparkfun AVC 2015 obstacles [2]

6

6

5

**Table 1**: Sparkfun AVC point system dependent on different time constraints

|  |  |  |
| --- | --- | --- |
| Time[s] | Speed[ft/s] | Points |
| 300 | 2.98 | 0 |
| 240 | 3.73 | 60 |
| 180 | 4.978 | 120 |
| 120 | 7.46 | 180 |
| 60 | 14.933 | 240 |

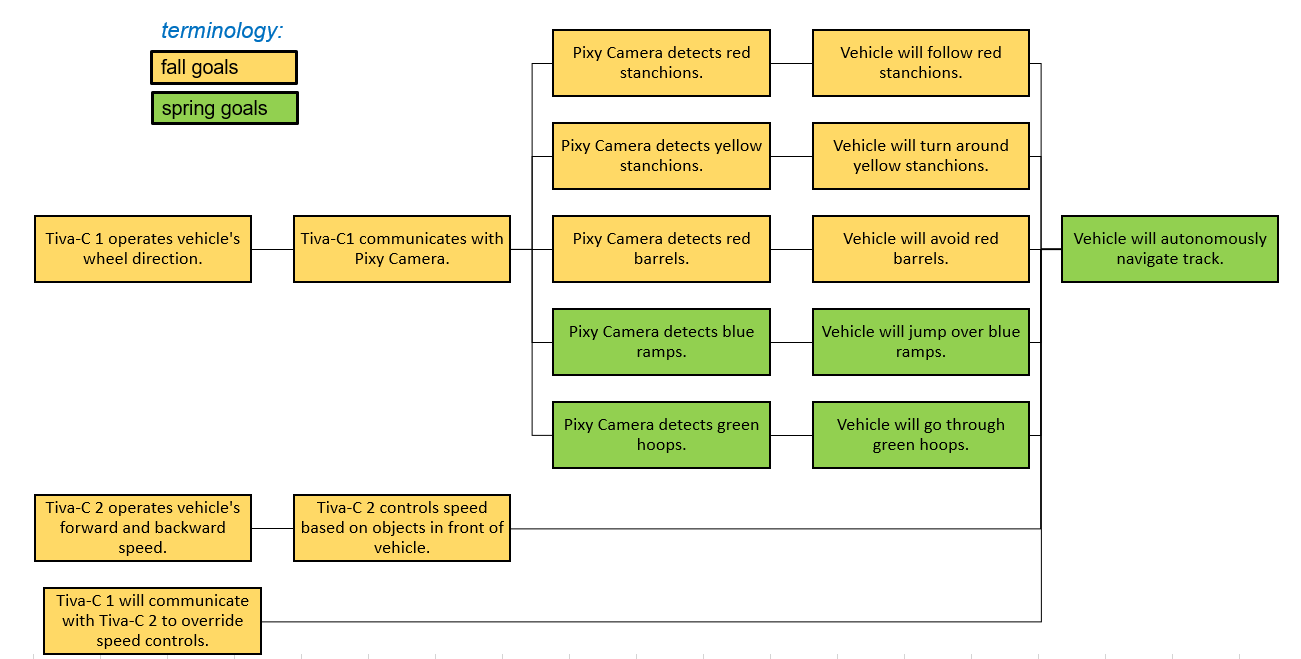
**Constraints**

The constraints we have encountered are the processing speed of the control system, vehicle speed, usage of GPS navigation and PIXY camera’s maximum frames per second. The competition is based off of a point system, to acquire the most points the vehicle must complete the track as quickly as possible, without use of GPS navigation, while interacting with the obstacles. One constraint that has been taken into consideration is the PIXY’s limitation of 50 frames per second, this means we have an updated image every 20[ms]. The constraint for an image to update, and be processed, is taken into account to find the optimal point for speed and 90% accuracy in object recognition.

# **Statement of Accomplishments**

The accomplishments thus far are, Tiva-C 1 operates vehicle’s wheel direction, and Tiva-C 2 operates vehicle’s forward & backward speed, communication between all the devices, and recognition of different objects. The test procedure for these goals include programming the microprocessors to oscillate the direction of the wheels and the speed of the motor. The tasks that are completed are crossed In Figure 8 below.

7



**Figure 8:** Goal Analysis with crossed out task completed for Sparkfun AVC 2016 project

On the track, objects are placed close together or further away depending on the part of the track the vehicle is on. The stopping distance or recognition of the object is important for the navigation of the track. To measure the stopping distance a brick was used as a placeholder, first the car was moved 50ft from the brick, then the car was accelerated to maximum velocity. When the car crossed the brick, full reverse was applied to stop the vehicle. Once the vehicle slowed to a complete stop, the distance was measured between the brick and the front of the car. Data collected can be seen in Figure 9. The average stopping distance for all the trials is 22.24 [ft] with a standard deviation of 2.06[ft].

8

**Figure 9**: stopping distance trials. Error bars on the graph display 1 standard deviation

The results of the stopping distance trials have led to a desire to obtain a better sensor such as LIDAR, or a longer range distance sensor. A better sensor will allow the vehicle to see further thus being able to stop sooner. Because the current sensors can only see up to 5 [ft], the top speed will have to be limited such that the vehicle has a stopping distance of 5 [ft]. LIDAR (Light Detection and Ranging) works by illuminating the object with a focus light and calculating the time it takes for the light pulse to bounce back. [1] This will provide a quicker detection than the infrared which uses a dispersed light.

Another measurement that was tested was the turning radius of the vehicle. To see how the turning radius would affect the control while turning. To measure the turning radius, the vehicle was driven in a circle around the operator then, once stopped, the distance from the operator to the vehicle was measured. Table 2 shows the results of this test.

9

## 

Table 2: trails of vehicle turning radius

To determine how the Tiva-C’s must control the RC car, measurements were taken while controlling the vehicle with the remote control. The measurement devices used were both an oscilloscope and a DC meter. Measurements can be seen in Table 3 below. It was found that all signals were a specialized type of pulse width modulation with a peak-to-peak voltage of 3.3[V], a period of about 10 [ms], and a neutral state (no action from motor or axel servo) of about 15% duty cycle. The data collected can be used to set limits for speed and direction via software.

Table 3: PWM measurement of accelerator, steering and trim setting

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Accelerator |  |  | Steering |  |  | Trim Setting | |
| State | Oscilloscope | DC Meter | State | Oscilloscope | DC Meter | State | DC Meter |
| Forward | 1.976 [ms] | .66 [V] | Right | 1.973 [ms] | .64 [V] | Right | .52 [V] |
| Neutral | 1.484 [ms] | .5 [V] | Neutral | 1.465 [ms] | .48 [V] | Neutral | .48 [V] |
| Reverse | 990 [µs] | .34 [V] | Left | 983 [µs] | .32 [V] | Left | .44 [V] |
| Peak-to-Peak | 3.276 [V] | NA | Peak-to-Peak | 3.276 [V] | NA |  |  |
| Period | 9.87 [ms] | NA | Period | 9.87 [ms] | NA |  |  |

The communication of choice between all the devices is I2C, communicating between devices is needed to ensure navigation and data collection is achieved. The information that is being transmitted between the devices is speed, object, dimensions of the object and the position, on the image, the object is located. The PIXY data returns height, width, color, and center of the object. Using the PIXY, the vehicle can differentiate a red barrel and a red stanchion, it follows the red stanchion and avoids the red barrel, by adjusting the direction of the wheels. The semester accomplishments are communication between the devices, Tiva-C control of speed & wheel direction, and recognition of red barrels and red stanchions.

An unexpected, and recent, change in design has taken place. After discussing the design with our facilitator it was decided to continue developing the vehicle without the use of the forward control. This is due to the fact that in a race, the optimal speed is the fastest, so by limiting the speed we are greatly hindering our performance. With that being said the PIXY camera is being used to determine directional decisions, while the speed is remaining constant. To accomplish this a proportional differential controller, or PD controller, is currently being implemented to allow the vehicle to interact with objects by changing the direction of the wheels alone. Once the controller is implemented the forward speed controller will be re-implemented to aid the successful passing of objects.

**Engineering Standards**

**ISO/IEC/IEEE 60559:2011**

Specifies formats and methods for floating-point arithmetic in computer systems. Recommendations for data exchange between devices.

**IEC 796-1:1990**

Standardization of microprocessor systems; and of interfaces, protocols, architectures and associated interconnecting media for information technology equipment and networks.

**Budget**

10

11

Table 4: Total Expenditures for Fall

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Labor** | | | | |
| Laborer | Price Per Unit | # of Units | Projected Total | Expended Total |
|  |  |  |  |  |
| Don Nguyen | 75.00 | 85 | 6,375.00 | 750.00 |
| Brandon Champagne | 75.00 | 85 | 6,375.00 | 750.00 |
| Aaron Zamora | 75.00 | 85 | 6,375.00 | 750.00 |
| Dr. John Glover (Consultant) | 150.00 | 40 | 9,000.00 | 600.00 |
|  |  |  |  |  |
|  |  | **Labor Total** | 28,125.00 | 2,850.00 |
|  |  |  |  |  |
| **Hardware** | | | | |
| Part | Price Per Unit | # of Units | Projected Total | Expended Total |
|  |  |  |  |  |
| Tiva-C microcontroller | 19.99 | 2 | 39.98 | 39.98 |
| Arduino Mega microcontroller | 38.00 | 1 | 38.00 | - |
| Pixy Camera | 69.00 | 1 | 69.00 | 69.00 |
| Li-Po Battery Charger | 25.00 | 1 | 25.00 | 21.10 |
| Voltage regulator | 15.00 | 1 | 15.00 | - |
| Accelerometer | 27.95 | 1 | 27.95 | 27.95 |
| Infrared Proximity Sensor Long Range - Sharp GP2Y0A02YK0F | 14.95 | 2 | 29.90 | 29.90 |
| Infrared Proximity Sensor Short Range - Sharp GP2Y0A41SK0F | 13.95 | 2 | 27.90 | 27.90 |
| Shock springs | 5.00 | 4 | 20.00 | - |
|  |  |  |  |  |
|  |  | **Hardware Total** | 292.73 | 215.83 |
|  |  |  |  |  |
| **Grand Total** |  |  | 28,417.73 | 3,065.83 |

Table 5: Total Expected Expenditures for Spring

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Labor** | | | | |
| Laborer | Price Per Unit | # of Units | Projected Total | Expended Total |
|  |  |  |  |  |
| Don Nguyen | 75.00 | 90 | 6,750.00 | 750.00 |
| Brandon Champagne | 75.00 | 90 | 6,750.00 | 750.00 |
| Aaron Zamora | 75.00 | 90 | 6,750.00 | 750.00 |
| Dr. John Glover (Consultant) | 150.00 | 40 | 9,000.00 | 600.00 |
|  |  |  |  |  |
|  |  | **Labor Total** | 28,125.00 | 2,850.00 |

Risks

Obviously the unexpected changes in the overall design pose a large risk, and in fact change the overall deliverables. But because a large portion of our design has changed we can now focus together, as a group, in completing the new functionality. Another risk is the destruction of equipment during testing. Because the equipment being used is sensitive and the goal of our project is to accomplish high speeds, the risk of damages is high. So to mitigate this risk all tests are currently being done with the vehicle propped up, off the ground.

# **Conclusion**

The purpose of the project is to bring recognition to the Cullen College of Engineering by winning the competition. For the conclusion of the fall semester the accomplishments are communication between the devices, Tiva-C control of wheel direction, and recognition of red barrels and red and yellow stanchions. The proposed design included the proximity distance sensor, after the stopping distance trial where performed, the average stopping distance of 22.4[ft] was recorded. In conclusion of the trial, we have decided that a LIDAR sensor would have a better performance than the distance sensor. The Pixy camera was taught to recognize red barrels and red and yellow stanchions. A revised design has led to a single Tiva-C controlling the direction of the front wheel based on the objects detect by the PIXY camera. During the competition points are awarded for interacting with objects along the course, and completing the course in under 5 minutes without the use of GPS. Completed objectives include both microprocessors operating the vehicle’s basic functions, direction, motor controls, communication between the Tiva-c and PIXY. Because of the revision, original goals have not been met, but new goals have. Going into the spring semester work on the vehicle recognizing the green hoop and blue ramp as well control algorithms for controlling the speed of the vehicle.

12

13

# Bibliography

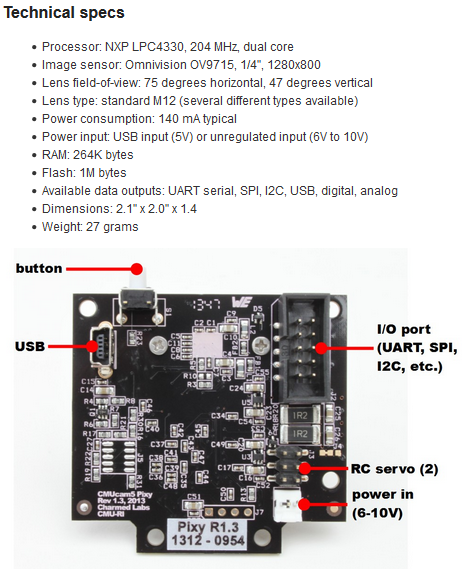
|  |  |
| --- | --- |
| [1] | http://www.lidar-uk.com/how-lidar-works/. |
| [2] | “. h. SparkFun Electronics. |
| [3] | “. 2. h. SparkFun Electronics. |
| [4] | "http://standards.ieee.org/findstds/standard/802.15.1-2005.html," [Online]. |

**Appendices**

14

**Appendix A: Pixy Specs**

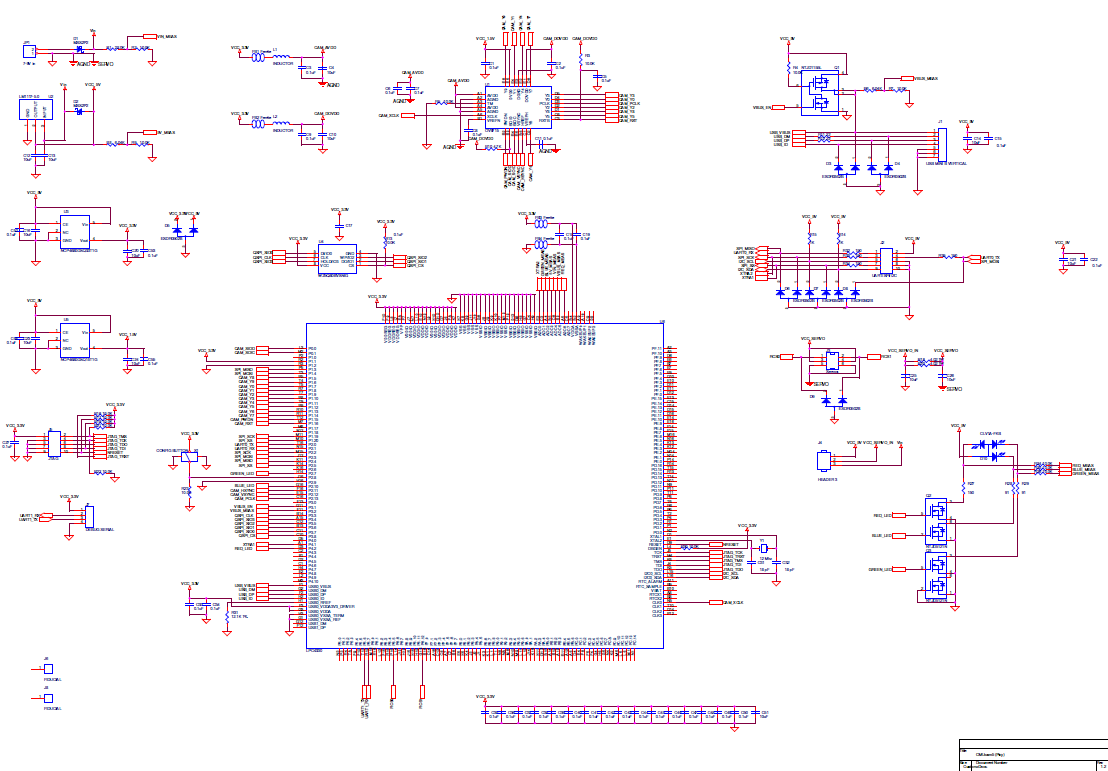
Figure 1 details the specifications of the Pixy camera



**Figure 1**: Pixy specifications

**Appendix B:Pixy Electrical wiring diagram**

Figure 1 below is the electrical diagram of the pixy



**Figure 2**: Pixy electrical diagram

**Appendix C: Communication code for I2C**

Appendix C include the communication driver that was written for the device to communicate with the Tiva-C.

#include "I2C\_Pixy.h"

//---------------------------------------------------------------------------

// I2C\_Pixy\_init(void) - Inits I2C0 module

// PB2 - SCL (Pixy 5), PB3 - SDA (Pixy 9)

//---------------------------------------------------------------------------

void I2C\_Pixy\_init(void) {

    /\*---- Enable I2C0, using Port B  -----------------\*/

    SysCtlPeripheralEnable(SYSCTL\_PERIPH\_I2C0);

    SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOB);

    //reset module

    SysCtlPeripheralReset(SYSCTL\_PERIPH\_I2C0);

    // Select the I2C function for these pins.

    GPIOPinTypeI2CSCL(GPIO\_PORTB\_BASE, GPIO\_PIN\_2);

    GPIOPinTypeI2C(GPIO\_PORTB\_BASE, GPIO\_PIN\_3);

    // Configure the pin muxing for I2C0 functions on port B2 and B3.

    GPIOPinConfigure(GPIO\_PB2\_I2C0SCL);

    GPIOPinConfigure(GPIO\_PB3\_I2C0SDA);

    // Enable and initialize the I2C0 master module.  Use the system clock for

    // the I2C0 module.  The last parameter sets the I2C data transfer rate.

    // If false the data rate is set to 100kbps and if true the data rate will

    // be set to 400kbps.

    I2CMasterInitExpClk(I2C0\_BASE, SysCtlClockGet(), true);

    //clear I2C FIFOs

    HWREG(I2C0\_BASE + I2C\_O\_FIFOCTL) = 80008000;

}

//---------------------------------------------------------------------------

// I2CReceive(uint32\_t slave\_addr, uint8\_t reg)

// read specified register from slave device

//---------------------------------------------------------------------------

uint32\_t I2CReceive(uint32\_t slave\_addr, uint8\_t reg) {

    //specify that we are writing (a register address) to the

    //slave device

    I2CMasterSlaveAddrSet(I2C0\_BASE, slave\_addr, false);

    //specify register to be read

    I2CMasterDataPut(I2C0\_BASE, reg);

    //send control byte and register address byte to slave device

    I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_BURST\_SEND\_START);

    //wait for MCU to finish transaction

    while (I2CMasterBusy(I2C0\_BASE))

        ;

    //specify that we are going to read from slave device

    I2CMasterSlaveAddrSet(I2C0\_BASE, slave\_addr, true);

    //send control byte and read from the register we

    //specified

    I2CMasterControl(I2C0\_BASE, I2C\_MASTER\_CMD\_SINGLE\_RECEIVE);

    //wait for MCU to finish transaction

    while (I2CMasterBusy(I2C0\_BASE))

        ;

    //return data pulled from the specified register

    return I2CMasterDataGet(I2C0\_BASE);

}

//---------------------------------------------------------------------------

// getWord()

// returns two byte word from two one byte values from I2C

//---------------------------------------------------------------------------

uint16\_t getWord(void) {

// this routine assumes little endian

    uint16\_t w;

    uint8\_t c;

    c = I2CReceive(PIXY, 0x00);

    w = I2CReceive(PIXY, 0x00);

    w <<= 8;

    w |= c;

    return w;

}

//---------------------------------------------------------------------------

// getStart() - defines normal or color code block, finds two consecutive sync words

// returns 0 - if no data

// returns 1 - if data is found

//---------------------------------------------------------------------------

int getStart(void) {

    uint16\_t w, lastw;

    lastw = 0xffff; // some inconsequential initial value

    while (1) {

        w = getWord();

        if (w == 0 && lastw == 0) {

            return 0; // in I2C and SPI modes this means no data, so return immediately

        } else if (w == PIXY\_START\_WORD && lastw == PIXY\_START\_WORD) {

            g\_blockType = NORMAL\_BLOCK; // remember block type

            g\_skipStart = 1;

            return 1; // code found!

        } else if (w == PIXY\_START\_WORD\_CC && lastw == PIXY\_START\_WORD) {

            g\_blockType = CC\_BLOCK; // found color code block

            g\_skipStart = 1;

            return 1;

        } else if (w == PIXY\_START\_WORDX) { // this is important, we might be juxtaposed

            I2CReceive(PIXY, 0x00); // we're out of sync! (backwards) - pull one byte out to resync 2-byte word

        }

        lastw = w; // save

    }

}

//---------------------------------------------------------------------------

// getObstacles() - Parses data block into Obstacle, assumes getStart() has been called

// maxBlocks is # of blocks to parse

// Returns # of blocks

//---------------------------------------------------------------------------

uint16\_t getObstacles(uint16\_t maxBlocks) {

    uint8\_t i;

    uint16\_t w, blockCount, checksum, sum;

    Obstacle \*obstacle;

    // g\_skipStart check to see if we have already read a sync

    // if not, check to see if there is no data and if so return zero

    // else, clear g\_skipStart

    if (!g\_skipStart) {

        if (getStart() == 0)

            return 0;

    } else {

        g\_skipStart = 0;

    }

    // iterate for each block received up to maxBlocks OR PIXY\_ARRAYSIZE

    for (blockCount = 0;

            (blockCount < maxBlocks) && (blockCount < PIXY\_ARRAYSIZE);) {

        checksum = getWord();

//      if (blockCount == 1) {

//          g\_skipStart = 0; // this is just to catch scan on second obstacle

//      }

        // Parses for sync byte. Checks whether this is normal or color code block.

        if (checksum == PIXY\_START\_WORD) // we've reached the beginning of the next frame, acts as getStart

        {

            g\_skipStart = 1;

            g\_blockType = NORMAL\_BLOCK;

            return blockCount;

        } else if (checksum == PIXY\_START\_WORD\_CC) {

            g\_skipStart = 1;

            g\_blockType = CC\_BLOCK;

            return blockCount;

        } else if (checksum == 0) {

            return blockCount;

        }

        // create local pointer to global obstacle

        obstacle = g\_obstacles;

        // Calculate checksum & save data

        for (i = 0, sum = 0; i < (sizeof(Obstacle) / sizeof(uint16\_t)) - 2;

                i++) {

            w = getWord();

            sum += w;

            switch (i) {

            case 0:

                obstacle[blockCount].color = w;

                break;

            case 1:

                obstacle[blockCount].x = w;

                break;

            case 2:

                obstacle[blockCount].y = w;

                break;

            case 3:

                obstacle[blockCount].width = w;

                break;

            case 4:

                obstacle[blockCount].height = w;

                break;

            case 5:

                obstacle[blockCount].distance = 0xff;

                break;

            case 6:

                obstacle[blockCount].objType = 0xff;

                break;

            default:

                break;

            }

        }

        // check checksum & if valid, increase blockCount

        if (checksum == sum) {

            blockCount++;

        } else {

            SCIUSB\_OutString("checksum error!\n");

        }

        // Next block sync, if not sync, end of data

        w = getWord();

        if (w == PIXY\_START\_WORD) {

            g\_blockType = NORMAL\_BLOCK;

        } else if (w == PIXY\_START\_WORD\_CC) {

            g\_blockType = CC\_BLOCK;

        } else {

            return blockCount;

        }

    }

}

//---------------------------------------------------------------------------

// printObstacle(Obstacle \*obstacle - prints to UART0 the obstacle

//---------------------------------------------------------------------------

void printObstacle(int index) {

    SCIUSB\_OutString("Object: ");

    SCIUSB\_OutUHex(g\_obstacles[index].objType);

    SCIUSB\_OutString("\n\r");

    SCIUSB\_OutString("\tcolor: ");

    SCIUSB\_OutUHex(g\_obstacles[index].color);

    SCIUSB\_OutString("  ");

    SCIUSB\_OutString("x: ");

    SCIUSB\_OutUHex(g\_obstacles[index].x);

    SCIUSB\_OutString("  ");

    SCIUSB\_OutString("y: ");

    SCIUSB\_OutUHex(g\_obstacles[index].y);

    SCIUSB\_OutString("  ");

    SCIUSB\_OutString("width: ");

    SCIUSB\_OutUHex(g\_obstacles[index].width);

    SCIUSB\_OutString("  ");

    SCIUSB\_OutString("height: ");

    SCIUSB\_OutUHex(g\_obstacles[index].height);

    SCIUSB\_OutString("  ");

    SCIUSB\_OutString("distance: ");

    SCIUSB\_OutUHex(g\_obstacles[index].distance);

    SCIUSB\_OutString("\n\r");

}

uint16\_t getColor(Obstacle \*obstacle)

{

    return obstacle->color;

}

uint16\_t getX(Obstacle \*obstacle)

{

    return obstacle->x;

}

uint16\_t getY(Obstacle \*obstacle)

{

    return obstacle->y;

}

uint16\_t getWidth(Obstacle \*obstacle)

{

    return obstacle->width;

}

uint16\_t getHeight(Obstacle \*obstacle)

{

    return obstacle->height;

}

uint16\_t getDistance(Obstacle \*obstacle)

{

    return obstacle->distance;

}

uint16\_t getObjType(Obstacle \*obstacle)

{

    return obstacle->objType;

}