MOBILE LIDAR 3D SCANNING INSTRUMENT

ECE 540: Final Project

Fall 2021

## David Nwagbuo, nwagbuo@pdx.edu

## Alex Beaulier, beaulier@pdx.edu

## Neamen Asfaha, neamen.asfaha@pdx.edu

**Table of Contents:**

[**Introduction:**](#_heading=h.tyjcwt) **3**

[**Main Objectives or outcomes of the Project:**](#_heading=h.3dy6vkm) **3**

[**Team Planning**](#_heading=h.1t3h5sf) **3**

[**Description of the project:**](#_heading=h.4d34og8) **4**

[**Questions for instructor:**](#_heading=h.2s8eyo1) **4**

# 

# **Introduction:**

The objective of this project is to create a 3 dimensional image using a garmin lidar lite sensor using a small rotational gimbal and an lcd display of the status. The team will integrate multiple components, IP from opencores and combine them into a mobile instrument.

# **Main Objectives or outcomes of the Project:**



Project visualization example

**Figure 1:** *Example 3d scan of city from LIDAR. XYZ data converted to visualize depth using an RGB scale of Z position.*

# **Team Planning:**

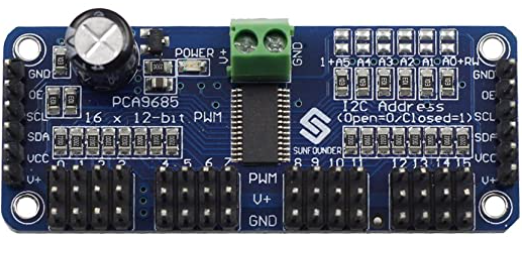
| **Task #** | **Due Dates** | **David’s Task** | **Alex’s Task** | **Neamen’s Task** |
| --- | --- | --- | --- | --- |
| Initial Proposal | 10-Nov | Initial study on the project topic. | | |
| Proposal Finalization | 14-Nov | Finalize the scope of the project after all Q/A and feedback from instructors and have this table filled in. | | |
| 1st half of the project | 21-Nov | * Have Lidar scanner fully powered on and able to get data readings. * Set up peripherals to use vivado (I2C protocol) | * Should have gimbal/servo motor powered on and functional * Set up peripherals to use vivado (PWM, assist with I2C) | * At this point the LCD display should be fully functional and able to display test texts. * Set up peripherals to use vivado (I2C protocol) |
| 2nd half of the project | 28-Nov | * Should be able to somewhat translate data readings and output it to VGA display | * Gimbal movement in all direction implemented and also push buttons setup | * Helping with integrating the different components, servo scan percentage complete and direction |
| Final Report and presentation | 5-Dec. Date of presentation to be finalized. | * Integrating all components together * Implementing data translation algorithms needed | * Integrating all components together * Merging individual part function codes into same project | * Integrating all components together |

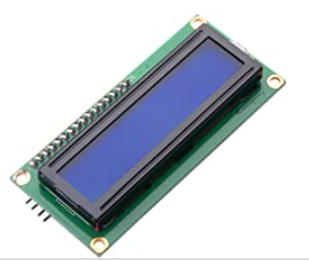
# **Description of the project:**

The team will create user selected scans from a LIDAR based scanning tool. The tool will pivot angular scanning on a constructed gimbal in the yaw(left to right) and pitch(up and down) directions. This will be used to acquire xyz points of the surrounding area up to 40m relative to the base. The points will be converted into pixels displayed on a VGA monitor. Image will be displayed in grayscale and pixel depth will be interpreted by a linear scale of pixel color through LIDAR data conversion. LEDs on the FPGA will update the progress of the scan. Pushbuttons will allow the user to change the scan direction from left, forward or right. A reset button will reset the system to stop scanning. An LCD display will show the current progress and selected direction. The LiPO battery will allow the system to be hand held and use buck converters for stepping down the voltage to peripheral equipment. A bluetooth module will also be used to transmit data to an Arduino as a precursor for future IOT development. Example parts given in proposal Appendix.

PARTS ACQUIRED:

****

****

****

**LiPo battery Buck converter LCD Display**

# **Questions for instructor:**

Is this a large enough scope?

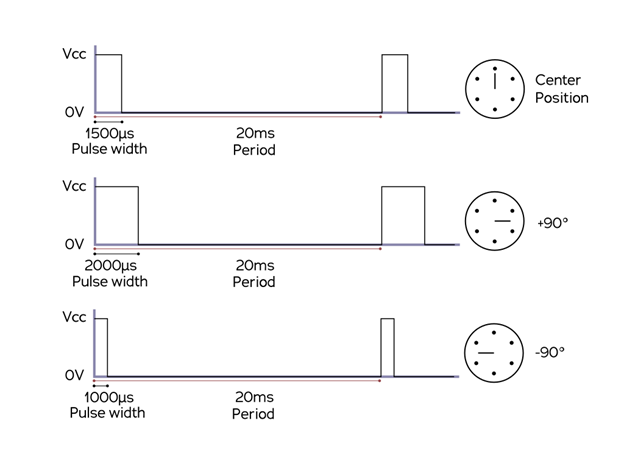
Maybe too much scope. Make sure you can make small interim goals that can be achieved.

**Servo Control Theory of Operation**

**Servo Background:**

Servos are small hobby motors controlled by pulse-width modulation (PWM). FPGAs support precise timing on outputs which can output PWM. Servos use the duration of the high period of PWM for position. The servos used in our project have a specification range of 180 degrees though manual testing shows the range is more closely to 270 degrees.. The time between pulses equates to a new position. The servo samples the duration of the high period and this interval is important because it determines the angular position. The majority of servos expect a pulse around 20ms. This is due to being from original signal encoding in PPM(pulse-position modulation) and that servos can only seek as fast as 20ms. RC servos run on 5V whereas the FPGA output is 3.3V. Therefore, buck converters will step down the 12V battery to 5V and will not be supplied from the FPGA. The neutral setting angle is approximately 1.5ms pulse width. The maximum angle is usually at 2.0ms and the minimum angle at 1.0ms. [1][2]

An example of the positioning scheme is seen below:



**Figure 1**: *Servo positioning scheme. [1]*

**Servo Register Setup**

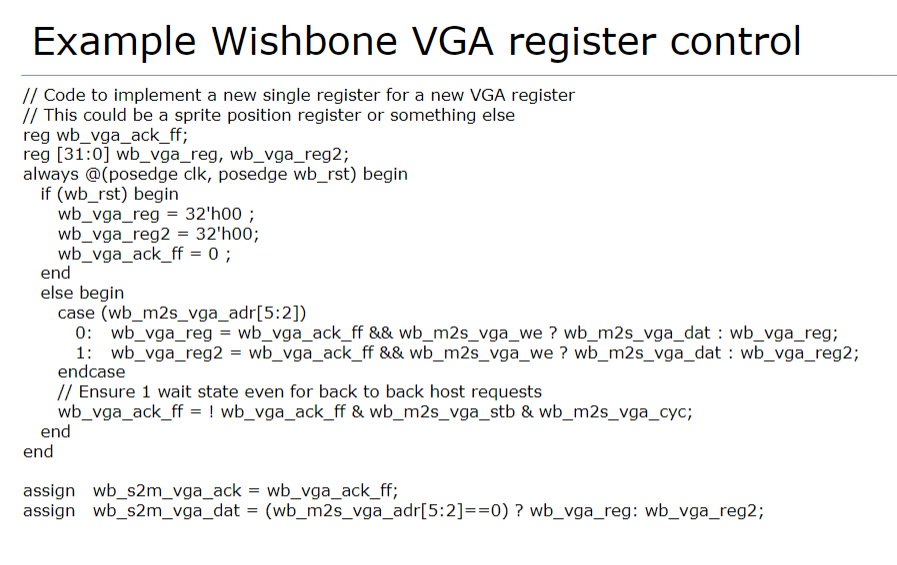
The register setup will follow as the initial VGA lab code with two additional registers instead of one. The example code setup for two registers is shown below. The servo registers will be stored within the VGA address space. 

Figure x: Example wishbone code for two registers.

**Servo Operation Layout**

The system operation will use System Verilog Code and C. C will be used to generate the movement loops with an angle request written to yaw and pitch addresses. System Verilog code will be used strictly for updating the register’s angle. It will generate the pwm clock signal, reset on system resets and update the position as PWM output through the GPIO of the Nexys Pmods. Our configuration uses Pmod D pins 0 and 1, D4 and D3.

Another aspect for this System Verilog setup requires multiple calculations and logic with descriptions listed below. These are necessary for operation and could be parameterized later in the model.

**System Verilog Skeleton Layout**

1. Two addressable registers that hold the duty cycle(angle request) of yaw and pitch. These will be fed into the pwm generator top module for updating the output pins.

2. Period generator – For PWM generation, the system clock of 100MHz is used. It should reset on reaching 20ms or a reset high. To convert this to a 20ms period using a counter:

a. Counter = (System CLK:100MHz)/(1/(Refresh:20ms \* 10^-3) = 20'd2000000

3. Duty Cycle update (Angle request input to System Verilog PWM generator)

a. When the counter is less than the duty cycle, the pin must be high, else drive the pwm low.

**C Code Skeleton Layout**

1. Loop Angle Yaw and Pitch Requests

a. Pitch will use 60 deg over 480 vertical scans

b. Yaw will use 60 deg over 640 horizontal scans

c. Current % of pitch scan will be used (pitch/480) to update the LEDS as a % of the scan completed.

d. LIDAR data will be requested in this loop, if LIDAR data is 0, a new data is requested and the angle will not continue. Could also take in 0 however the color map would be very skewed.

e. The LIDAR data(depth) threshold will be converted to RGB. Blue is close, Green is medium and Red is far away. Each color has 16 bits. Depth will be partitioned according to 40m/3 = 13.333 / 16 bits of color. Case statements can specify these ranges for the RGB output.

i. Alternatively the depth could be partitioned based on the minimum and maximum depth in the scan allowing for greater resolution within the data.

ii. Due to the maximum resolution of the VGA being 16 bits, the team opted to control further depths with the full three colors rather than segment the height by RGB as seen in the proposal.

f. Data should be written out to the LCD

2. Angle converter – The angle requested based on the current VGA will be 180 degrees / 3 different ranges. All three scanning ranges, left middle center, will therefore have 60 degrees and need to represent a 640 horizontal pixel range and 480 vertical pixel range. This will be a floating point such as 100.5\* whereas the register can only hold 32 bit values and could provide difficulty transferring to the pwm generator. The angle needs to be converted to a frequency between 1ms and 2ms. This poses an issue if the floating point is passed into the register and System Verilog may have difficulty interpreting. Therefore, this calculation will be performed in C and written to the register as a 32 bit frequency as the duty cycle request.

a. Assumptions

i. Note the calculation in iii uses floor so there is an absolute rounded value that can be read by System Verilog

b. Theoretical Angle Request

i. Degrees = 0 to 180

ii. Partitioned total, 1ms range = 20'd1000000 = 1,000,000

iii. Floor(parition total / degree total) = partition per degree

iv. Input Request Degree \* partition per degree = duty cycle

**Servo Verification**

The verification for this part of the project will be through System Verilog DUT within Vivado, oscilloscope verification of the PWM output, and afterwards physical verification of each servo.

**System Verilog DUT Image**

**Oscilloscope verification Image**

**Physical Video of Servo Movement**

**Discussion and Adjustments To Plan**

XYZ doesn’t work as thought, adjusted frequency….

**Servo Conclusions**

The servo works as requested, noticed xyz features differed slightly.

**Appendix A**

References:

[1]<https://en.wikipedia.org/wiki/Servo_control>

[2]<https://www.thomasnet.com/articles/instruments-controls/servo-motor-controllers/>

[3]<https://www.electronicshub.org/esp32-servo-control/>