# LIDAR Autonomous Mapping System

Trent Bennett
*University of Utah*
*Computer Engineering*
Salt Lake City, UT

Ty Jensen
*University of Utah*
*Computer Engineering*
Salt Lake City, UT

Kristopher Wolff
*University of Utah*
*Computer Engineering*
Salt Lake City, UT

***Abstract*—The Autonomous LIDAR Mapping System was designed and built as a custom 3D modeling mapping prototype used for many different applications like surveying, construction, or architectural purposes. This system takes advantage of LIDAR technology to map or scan static environments for applicable point cloud data and extract a 3D model object of a given space. This project eliminates the need for humans to measure manually, and presents it as a low cost solution. The custom built 3D modeling software has a simple user interface where the final 3D model image is displayed. This project is delivered as a prototype of an affordable LIDAR mapping system.**

## I. Introduction

THERE is great demand in the current market for efficient and affordable LIDAR mapping systems for indoor and outdoor environments within the architectural, engineering and construction industries. Technical advances in this field continue to grow with market projections of $4.4 billion by the year 2023 [1]. The time and cost savings for humans to use autonomous systems is inherently desired. It is attractive to many different sectors in the market. [2]. This work presents our design approach to construct such a prototype device to transform the final output into a 3D model object.

### A. Motivation

One major motivation for this project was the opportunity to build a custom mapping robot with the ability to provide information to create 3D models for historic building structures to protect, preserve, and archive for future generations. One example to this was the fire that destroyed the Gothic architecture of the Notre-Dame de Paris cathedral in Paris, France [3].

### B. LIDAR Technology

Light Detection and Ranging (LIDAR) technology is a reliable laser scanning technology that sends light pulses reflecting off objects and measuring distances of an obstacle [4]. LIDAR is an extremely useful tool for autonomous robotic platforms and a large part to our 3D modeling visualization scanning robot. This 3D modeling environment technology is an important topic for many research fields. Some of these applications include autonomous navigation like aerial vehicles (UAV's), self-driving cars, automated guided vehicles (AGV's), and mapping architectural buildings [4].

This project implements our custom version of what is referred to as the LIDAR Autonomous Mapping System or the acronym "LAMS". The actual cost of 3D LIDAR sensors are extremely high, which limited our ability to only be able to acquire a 2D LIDAR laser scanner. The core purpose for LAMS was to perform omnidirectional laser range finding for the surrounding environment [5]. This process adopted a laser triangulation ranging principle by operating high-speed vision acquisition and processing hardware [6].

### C. Project Scope and Overview

The originally proposed scope for this project was delivered as a custom terrestrial low-cost 3D modeling scanning robot prototype. We used the Slamtec RPLIDAR A2M8 2D 360-degree 12m radius laser scanner. It is mounted to the top of a static (terrestrial) tripod stable base as shown in Figure 1. All the components and brackets required were designed using CAD software and then 3D printed to secure the hardware to the device. The servo motor acts as the turning mechanism to the scanning assembly. It performs a 0 to 180 degree incremental turn on the x-y axis, while the LIDAR sensor gets mounted to the L-bracket and will rotate continuously at 360 degrees on the z-axis. This is meant to achieve the full 360 degree 3D cloud map using only a 2D scanner. It must be noted that our scans are limited to ranges of 12m radius as the max distance for the scanner, so there are limitations to this design.

To store the point cloud data from the scanner we built an embedded PCB design which includes SD card protocol in

Figure 1: LAMS robot on terrestrial tripod base

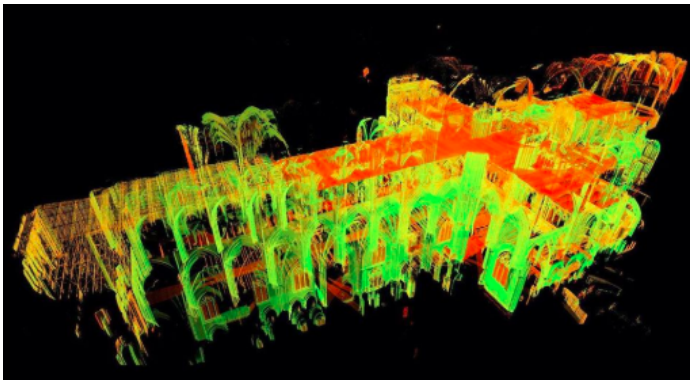


Figure 2: Example 3D laser scan of Notre Dame Cathedral in France

SDHC format (4-8 GB storage capacity) to handle and store the data. It gets stored as a .lams file into the SD card.

The custom 3D modeling software tools convert the .lams file from the SD card into an .obj file. Therefore, the first purpose of the software is to convert the data files for viewing. It can then display all the points in the 3D viewer from the converted obj file. The user can then use the visualization tools to look at the points (map) and its features to display on your computer. The final result is similar to Figure 2 of the point cloud image. All the details to the design and implementation are given in this report.

The final scope of this project was separated into the 3 major components, and satisfies what was originally proposed.

1) Designed and built a custom LIDAR mapping robot mounted to a terrestrial tripod base to scan for point cloud scanning.
2) Designed and built an integrated embedded PCB to store the data with SD card protocol (SDHC).
3) Designed and built a custom 3D modeling software to convert and produce the final 3D model images and files.

## II. Background

Light Detection and Ranging (LIDAR) mapping is used for many applications. Archaeological studies have used aerial LIDAR scans to locate and map Mayan ruins in the Mexican states of Tabasco and Chiapas. The LIDAR sensor is fixed to an aerial vehicle to map the ground below. These scans are able to penetrate the dense jungle foliage and provide mapping that researchers review. These LIDAR scans helped researchers discover ruins were much more extensive than previously thought [7].

In 2015 Andrew Tallon, a professor of Art History at Vassar College, used a LIDAR sensor mounted on a tripod to make high detail laser scans of the Notre Dame Cathedral (Figure 2). These LIDAR scans of Notre Dame are very highly valued ever since the fire in 2019 that destroyed the cathedral because of their possible use in the reconstruction effort [3]. Researchers at The University of Tubingen in Germany and Orebro University in Sweden have constructed a mobile robot that can navigate through a building and scan the space using LIDAR. The LIDAR scans are then compiled into 3D model that is useful for construction and renovation of an office or living space [8].

The LIDAR mapping of Notre Dame and autonomous mapping robots are similar to LAMS. Both of these implementations rotated a LIDAR scanner around the space to construct a point cloud data set which is processed into useful 3D models. LAMS is designed to produce a similar result, albeit, limited to being a prototype of these other systems.

| Qty | Part |
|---|---|
| 1 | Part: IC1 - ATSAME54N20A-AUT<br>Mfr: Microchip - MCU 120MHZ 1024KB FLASH 100 TQFP |
| 1 | Part: U1 - MAX20034ATIR/VY+<br>Mfr: Maxim Integrated - 2.2MHz, 36V, Dual Buck Control |
| 1 | Part: OSC1 - CX3225CA12000D0KPSC1<br>Mfr: Kyocera - Crystals 12MHz 8pF 3.2x2.5x0.8mm |
| 1 | Part: SD-CARD-CONN - GSD09002SEU<br>Mfr: Amphenol - Memory Card Connect SD, 9 pos, 15 |
| 2 | Part: RESET, START BUTTON - LL3301BF065QJ<br>Mfr: E-Switch - PB Switch 50mA 12VDC F065 9.5mm J-Lead |
| 8 | Part: C5,C6,C7,C8,C10,C11,C15,C16 - C0603C104K5RACTU<br>Mfr: KEMET Corp- 100nF 50V X7R 10% Pad SMD 0603 |
| 2 | Part: C1,C4 - C0603C472J5RACTU<br>Mfr: KEMET Corp- Cap Ceramic 4.7nF 50V X7R 5%Pad 0603 |
| 2 | Part: C2,C3 - 04025A4R7JAT2A<br>Mfr: AVX - Cap Ceramic 4.7pF 50V C0G 5% Pad SMD 0402 |
| 1 | Part: C9 - GRM188R61E106KA73D<br>Mfr: Murata Elect- MLCC - SMD/SMT 10UF 25V 10% 0603 |
| 2 | Part: C12,C13 - GRM1885C1H8R0CA01D<br>Mfr: Murata Elect- MLCC SMD/SMT 8pF 0603 |
| 1 | Part: C14 - GRM188R61C475KAAJD<br>Mfr: Murata Elect- SMD/SMT 0603 4.7uF 16V *Derate |
| 1 | Part: CBIAS - C1206C685K4RACTU<br>Mfr: KEMET Corp- Ceramic 6.8uF 16V X7R 10% Pad 1206 |
| 5 | Part: COUT1, CIN - C4532X7R1C336M250KC<br>Mfr: TDK - MLCC SMD/SMT 1812 16V 33uF X7R 20% |
| 5 | Part: COUT2 - GRM31CR71A226ME15L<br>Mfr: Murata- Cap 22uF 10V X7R 1206 20% 3.2mOhm 0.72nH |
| 2 | Part: D1,D2 - MBRS140T3G<br>Mfr: ON Semiconductor- Rect. Diode Sch Si 40V 1A 2-Pin |
| 1 | Part: LED STATUS - 155060VS75300<br>Mfr: Wurth Elektronik- WL-SMSW Grn 2V 20mA 572nm |
| 2 | Part: LED RX, LED TX - 155060VS75301<br>Mfr: Wurth Elektronik- WL-SMSW Ambr 2V 20mA 610nm |
| 1 | Part: L1 - SRP1238A-3R3M<br>Mfr: Bourns- Fixed Inductor 3.3uH 20% SMD 1238 AEC-Q200 |
| 1 | Part: L2 - MSS1260-682MLB<br>Mfr: Coilcraft- 6.8uH 20% 18.5mOhm 3.8Arms 20C 5.9Arms |
| 1 | Part: R1 - ERJ-2RKF1022X<br>Mfr: Panasonic- TF 0402 10.2K Ohm 1% 0.1W(1/10W) |
| 2 | Part: R3,R4 - ERJ2GEJ513X<br>Mfr: Panasonic- TF 0402 51K Ohm 5% 0.1W(1/10W) |
| 1 | Part: R2 - ERJ2RKF1052X<br>Mfr: Panasonic- TF 0402 10.5K Ohm 1% 0.1W(1/10W) |
| 1 | Part: R5 - ERJ-UP3F3300V<br>Mfr: Panasonic- TF 0603 0.25W 1% 330ohm AEC-Q200 |
| 5 | Part: R6,R8,R12,R13,R15 - ERJ-UP3F1002V<br>Mfr: Panasonic- TF 0603 0.25W 1% 10Kohm AEC-Q200 |
| 4 | Part: R7,R9,R10,R11 - ERJ-PA3F1003V<br>Mfr: Panasonic- TF 0603 100Kohm 1% Anti-Surge AEC-Q200 |
| 1 | Part: R14 - ERJ-PA3F1001V<br>Mfr: Panasonic- TF 0603 1Kohm 1% Anti-Surge AEC-Q200 |
| 3 | Part: R16,R17,R18 - ERJ-UP3J101V<br>Mfr: Panasonic- TF 0603 0.25W 5% 100ohm AEC-Q200 |

Table I: Hardware Bill Of Materials

## III. Hardware

Our project's hardware can be divided into 2 sections, integrating our components with our processor and developing the embedded system. The processor decision was heavily based on the necessity of using an SD card, so we decided to use Microchip's SAME54 series microprocessor. The decision resulted in the purchase of a SAME54 development board: the SAME54 Xplained Pro Evaluation Kit. This board allowed us to begin designing the embedded software before an embedded board would be developed to ensure working programming as we didn't want to risk a faulty PCB design.

### A. Component Integration

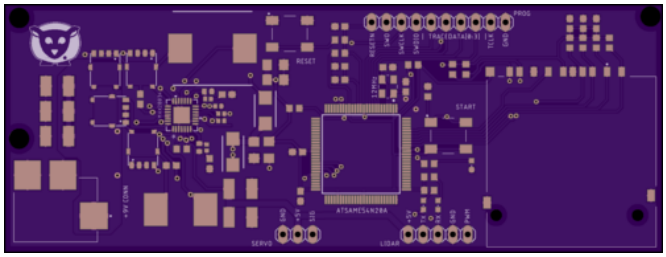
To integrate the components with our processor, we integrated each component, one-by-one into our software. To start off, we used internal timer peripherals to produce a PWM signal to the servo, UART communications to the LIDAR, as well as simple GPIO signals for a status LED and start button to start a scan.

The servo was an SPT5435 [9] that provided enough torque to rotate our platform without a lot of stress. The decision for this servo was also loosely based around a previous design that required more torque. We also wrote a small algorithm that allows us to give an angle and will adjust the timer peripheral to switch to its relative count specific to this servo. After working with the servo and seeing its performance, it was decided that if this were to go into production or this project was redesigned, a better solution would be to use a rotary motor for smoother angle transitions.

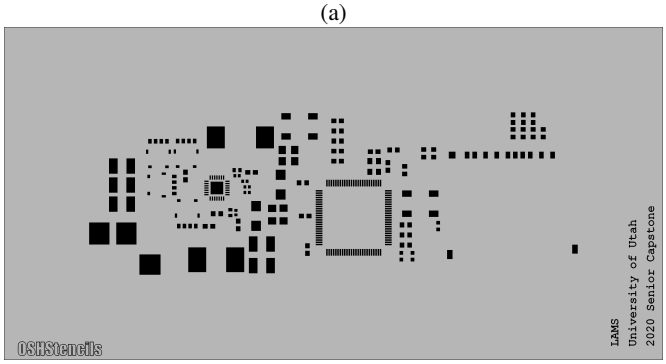
Next we worked to integrate the LIDAR sensor. After finding the sensor communication protocol [10], we were able to generate the required sequences of bytes to send to the LIDAR as well as receive and process the incoming data. This was the most challenging aspect of writing the software as much of the signals sent and received were time specific and needed to keep track of the bytes sent back and forth, as well as fully understanding the protocol in operation. To help our debugging process, we utilized a Saleae logic analyzer that allowed us monitor the signal traffic to ensure the bytes incoming and outgoing were what was expected.

Lastly we developed communications with the SD card. In order to save readable files, we utilized an open-source FATFS driver module [11] that allowed us to write file-specific data while also updating the base data of the SD card, keeping track of file structures and file management.

Once all basic communications had been tested and we were able to combine all the developed methods to be called within a single scan process. We decided that there should be a maximum number of scans per servo angle, as well as a maximum time spent on each angle in case of large rooms or incomprehensible objects that LIDAR laser could not detect. One such object was mirrors as the laser is not refracted directly off of the mirror, rather rebounding at a 90-degree angle from its position.



(a)




(b)

Figure 3: (a) Custom PCB stencil. (b) Custom PCB board.

### B. Embedded System

Designing the embedded system was a tricky task. We developed our boards and schematics using EAGLE where we carefully sifted through our component's different documentations and datasheets to ensure correct setup. Once our design components and boards arrived (Figure 3a), we used a PCB stencil (Figure 3b) for correct solder-paste placement and a reflow oven to melt the solder-paste, securing the components without damaging the microprocessor.

We wanted all voltages to run through the system to basically have a single power source that can control both the 3.3V power supply to the microprocessor while also outputting the 5V connections to the servo and LIDAR. We decided to use a voltage regulator capable of outputting the 2 voltages with an input power of 9V. However, this is what caused our custom PCB design to not work. After assembling the board, we found that the outputted voltages from the voltage regulator were not close enough to our desired voltage levels. We believe the reason for this was due to our attempt of low-cost development manufacturing, as well as the irregular component values needed from the voltage regulator to operate and output the required voltage levels. We now know of 2 safer options for powering a microprocessor: (1) We can use a variable voltage regulator to adjust voltages to desired levels or (2) we can use a voltage regulator that only drops the voltage to the desired 3.3V.

## IV. Software

### A. Software Language

The LAMS 3D Viewer was written completely in the programming language Java. We chose to write the viewer

in Java in order to make it cross platform for all major operating systems and for portability. As a result, our software runs from a single executable jar file on Linux, Mac, and Windows without any installation, assuming that the Java Runtime Environment is already installed.

### B. Software Structure

The Java code is split into two main packages. The first package is the .lam file to .obj file converter and the second is the 3D Viewer. Data stored on the SD card is formatted in spherical coordinates with the form (r, $\theta$, $\phi$). This must be converted to Cartesian coordinates before the data can be displayed in the LAMS 3D Viewer.

### C. LAM to OBJ Converter

The converter software converts the Spherical Coordinates to Cartesian Coordinates, generating edges between the points and a mesh with triangular faces. Because the model could have void areas (holes) where no LiDAR data is available, not all possible faces are valid. In order to ensure that the software can recreate the scanned area accurately, the number of $\theta$ and $\phi$ steps must be provided. For example, if the LAMS scan is returning 1 scan per angle, the $\theta$ steps would be 360 and the $\phi$ steps would be 180.

The converter software steps through every scan point to look for data at each $\theta$ and $\phi$ and creates edges between the adjacent points. If data is missing for a certain point, the converter leaves a hole in the model at that point. Once all the edges are created, the converter will create faces from the list of edges. Creating edges and faces from a point cloud requires a good deal of calculation. In order to make the processing easier, we created classes for faces, edges, spherical coordinates and Cartesian coordinates. These different classes have helper methods to assist in the processing of the point cloud.

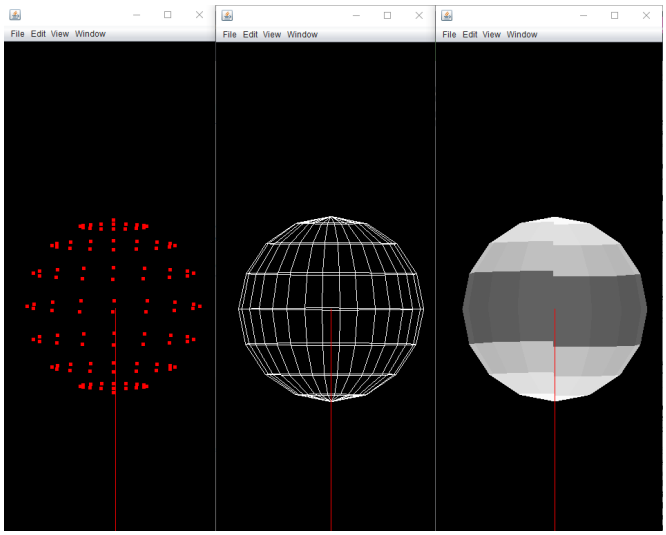



Figure 4: Vertex, wireframe and solid views of the LAMS 3D viewer side by side.

### D. LAMS 3D Viewer

The converter supports the LAMS 3D Viewer which will display the point cloud and model to the user and allow the user to make sense of and edit the data. The LAMS 3D Viewer has 3 views: the vertex view, the wireframe view, and the solid view as seen in Figure 4. The LAMS 3D Viewer has a drawing panel, which is the main feature of the Graphical User Interface (GUI), a menu bar, and satellite windows for tools and settings. The LAMS 3D Viewer has one class that handles all of the graphics processing. This class has methods to add points, lines, and faces to lists which will be drawn to the canvas. The Draw3D class will draw to the screen when the refresh method is called. Because there are no animated or moving objects, a static image rendered to the screen worked for our 3D viewer.

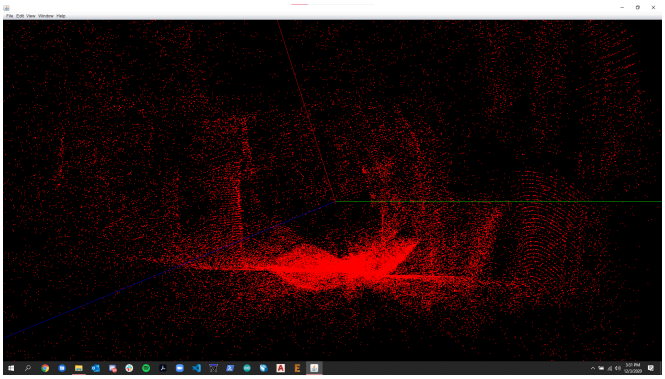
The LAMS 3D Viewer has a move and select tool in the toolbar. When the move tool is selected, the mouse click and drag will rotate the 3D Model. When the select tool is selected, the mouse click and drag will allow the user to highlight vertices which can be deleted by hitting the delete key. When only two points are selected, the 'E' key will create an edge between the two vertices. When 3 or 4 vertices are selected, hitting the 'F' key will create a face between the vertices. In both the selection mode and the move mode, the arrow keys the 'Q' key and the 'W' key can be used to navigate around the model.

The settings editor can be accessed from the view item in the menu bar. Aesthetic options for all 3 views of the 3D viewer are in this editor. These settings include: changing colors for all components in the editor, enabling the x, y, and z axes, enabling the xy-grid, and changing the size of points in the vertex view.

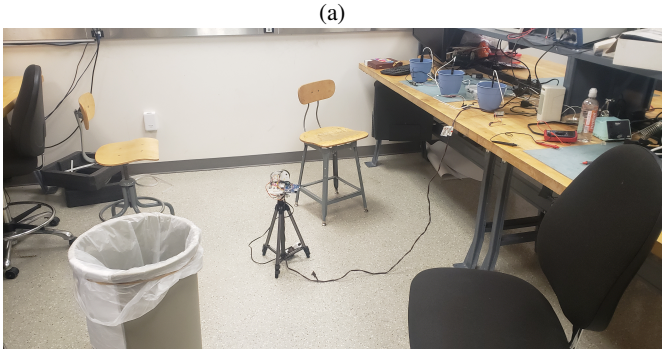
$$
\begin{aligned}
R_x(\theta) &= \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{vmatrix} \\
R_y(\theta) &= \begin{vmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{vmatrix} \\
R_z(\theta) &= \begin{vmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
x_p &= x * scale + screenWidth/2 \\
y_p &= y * scale + screenHeight/2
\end{aligned} \tag{2}
$$

The Draw3D class handles all the calculations required to project 3d points onto a 2D screen. Any 3D point projected to the screen must first be rotated and then be projected to the 2D screen space from the 3D world space. Equation 1 shows the rotation matrices that must be applied to the $< x, y, z >$ vector. After the points have been rotated in world space, the x and y components of the resulting vector are applied to Equation 2. After this transformation, the resulting points can be drawn to the draw panel of the 3D Viewer. Figure 5a shows the point



(a)



(b)

Figure 5: (a) LAMS scan example of the CE senior lab. (b) 3D viewer output of software from of the scan of the lab as shown in (a). Good example of a scan and the viewer together.

cloud representation of a LAMS scan with an environment shown in Figure in the LAMS 3D Viewer.

## V. Full Implementation

Ultimately, at the end of this project the completed LAMS system turned out to be extraordinarily similar to what was expected and planned in our proposal. The design implementation was configured more streamlined than what was first thought. We ended up using only one servo motor (only needed one) – because we took advantage of the continuously turning LIDAR sensor, allowing us to get the full 360 degree range we needed from only using a 2D sensor. This turned out to work in our benefit as the servo ended up having a slight vibration that would cause more inaccurate scans.

### A. Final Risk Assessment and Testing

One of the biggest challenges for the team during the testing phases was integrating and assessing bugs throughout the system. All the components were tested separately first, and then combined to work through some of the problems. From the first few scans, we needed to coordinate and work through the calculations of the data coordinates as given by the LIDAR scanner, so that the transfer of data was valid.

After our custom PCB was soldered, the final outputs were not fully realized. The hard work that was given for this board should not be lost however, giving a great experience in the process with the knowledge that we tried to build this ourselves from scratch. The final product can be seen in Figure 6a and Figure 6b.

(a)

(b)

Figure 6: (a) LAMS from overhead. (b) LAMS from the side.

### B. Low Cost Solution

Our proposal for this project was to finally deliver a low-cost LIDAR system prototype. This was fully realized as a typical LIDAR mapping system cost is in the ranges of $5000 to $85,000 in the current market. We completed this project for under $1000 (Table II). Therefore, the final result turned out to be successful as we completed the requirements for this project originally proposed.

| Part | Qty | Cost |
|---|---|---|
| LIDAR Scanner | 1 | $396 |
| Servo Motor | 1 | $30 |
| SAM E54 Xplained Pro Evaluation Kit | 1 | $75 |
| 3D Printer Filament | 1 | $25 |
| PCB Design/Ordering | 2 | $200 |
| Misc Components/Accessories | 1 | $45 |
| **Total Cost = $771** | | |

Table II: Final Cost of the LAMS Project

## VI. Conclusion

This project is of great importance to documenting 3D models of buildings and structures using LIDAR technologies. This system was built as a low-cost prototype solution for architects, engineers, and construction workers benefiting them with time and money to help map locations. The system makes for a more autonomous approach in getting 3D mapping and surveying information available. Indeed, there were some limitations to the project in general, but a success overall. The knowledge and experience gained will be a benefit for future opportunities.

The full integration of this prototype design incorporated a good mix of hardware and software concepts. The management and cooperation of team responsibilities were communicated and utilized mainly virtually because of COVID. All team functions were performed as a multi-disciplinary team for various roles. The final result was a success and ultimately delivered on the originally proposed project to meet the specifications. Overall, we are proud of the work that was accomplished for this project.

## References


[1] "Terrestrial Laser Scanning Market." [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/terrestrial-laser-scanning-market-3652955.html.

[2] A. Harchowdhury, L. Kleeman, and L. Vachhani, "3D Sensing of a Moving Object with a Nodding 2D LIDAR and Reconfigurable Mirrors," Tech. Rep., 2019. [Online]. Available: https://arxiv.org/abs/1912.13461

[3] A. Tallon, "Historian uses lasers to unlock mysteries of Gothic cathedrals," Apr 2019. [Online]. Available: https://www.nationalgeographic.com/news/2015/06/150622-andrew-tallon-notre-dame-cathedral-laser-scan-art-history-medieval-gothic/

[4] R. Athavale, D. H. Ram, and B. B. Nair, "Low cost solution for 3d mapping of environment using 1d LIDAR for autonomous navigation," *IOP Conference Series: Materials Science and Engineering*, vol. 561, p. 012104, nov 2019.

[5] J. Li, X. He, and J. Li, "2D LiDAR and camera fusion in 3D modeling of indoor environment," 2015.

[6] Z. Fang, S. Zhao, S. Wen, and Y. Zhang, "A Real-Time 3d Perception and Reconstruction System Based on a 2d Laser Scanner," *Journal of Sensors*, vol. 2018, pp. 1–14, 2018.

[7] A. F. Chase, D. Z. Chase, J. F. Weishampel, J. B. Drake, R. L. Shrestha, K. C. Slatton, J. J. Awe, and W. E. Carter, "Airborne LiDAR, archaeology, and the ancient Maya landscape at Caracol, Belize," *Journal of Archaeological Science*, vol. 38, no. 2, p. 387–398, Feb 2011.

[8] P. Biber, H. Andreasson, T. Duckett, and A. Schilling, "3d modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera," vol. 4. IEEE, 2004, pp. 3430–3435 vol.4.

[9] *SPT5435LV-180W*, 2020. [Online]. Available: http://www.spt-servo.com/Product/015234339.html

[10] *RPLIDAR Interface Protocol and Application Notes*, Slamtec, 2019. [Online]. Available: http://bucket.download.slamtec.com/ccb3c2fc1e66bb00bd4370e208b670217c8b55fa/LR001_SLAMTEC_rplidar_protocol_v2.1_en.pdf

[11] ChaN, *FatFS - Generic FAT Filesystem Module*, 2015. [Online]. Available: http://elm-chan.org/fsw/ff/00index_e.html