Implementation of Direction Cosine Matrix on a PSoC-5 Microcontroller for Robot Localization on Inclined Terrains

Garth Herman, Aleksander Milshteyn, Airs Lin, Manuel Garcia,
Charles Liu, Khosrow Rad, Darrel Guillaume, Helen Boussalis

Structures, Pointing, and Control Engineering (SPACE) University Research Center
College of Engineering
California State University, Los Angeles
5151 State University Drive
Los Angeles, CA 90032 USA
gherman@calstatela.edu

***Abstract*—** Robots facilitate exploration of hazardous environments during response to catastrophe. Autonomous robots involved in search and rescue operations require accurate orientation information to navigate uneven or inclined terrains. A Hybrid Routing Algorithm Model has been proposed by the Structures, Pointing, And Control Engineering (SPACE) University Research Center (URC) at California State University of Los Angeles. This model envisions three-layered terrain mapping with obstacle representations from various information sources such as satellites, UAVs, and onboard range sensors. The orientation information of a robot is obtained from a 9 degrees-of-freedom inertial measurement unit. In the absence of external magnetic fields, a magnetometer sensor is capable of outputting accurate heading information if it is firmly mounted on the robot chassis and its reference frame is parallel to the ground. This paper demonstrates an implementation of a direction cosine matrix (DCM) algorithm on a Cypress Programmable System on a Chip-5 (PSoC-5) platform to obtain correct orientation information for the semi-autonomous robot. This semi-autonomous robot is capable of navigating and mapping the surrounding environment. A PSoC-5 is utilized as a microcontroller to control the platform and acquire data from multiple onboard sensors. The DCM algorithm is demonstrated to obtain and maintain global reference system information for a mobile robot under varied terrain inclinations.

This work was supported in part by the NASA University Research Center Program under Grant NNX08A44A and National Science Foundation (NSF) Grant 230953.

G. Herman is with the Structures Pointing and Controls Engineering (SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (phone: 323-343-5445; e-mail: gherman@calstatela.edu).

A. Milshteyn is with the Structures Pointing and Controls Engineering (SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (e-mail: amilsht@calstatela.edu).

A. Lin is with the Structures Pointing and Controls Engineering (SPACE) University Research Center at California State University of Los Angeles, Los Angeles, CA 90032 USA (e-mail: airslin@hotmail.com).

***Keywords*—** *system-on-chip, mobile robot, Direction Cosine Matrix, IMU, inclined surfaces, semi-autonomous, data acquisition, environment-mapping.*

I. Introduction

ADVANCES in robotics technology increase the utility of diversely applied platforms. Robots mitigate risk to emergency responders, and facilitate exploration of dangerous environments. These applications require information about the position and orientation (localization) of a robot. Localization information is crucial if a robot is used to locate victims and hazards of a catastrophe, or explore inhospitable or extraterrestrial environments. Accurate localization information is required to optimally navigate a robot through an environment [1].

Robots have been successfully deployed for emergency response and extraterrestrial exploration. A robotic platform was used to investigate radiation levels after a tsunami decimated a Japanese nuclear reactor on May 11, 2011. The robot was tested and retrofitted to ascertain its efficacy withstanding the anticipated conditions within the reactor before it was deployed. The localization of the robot was tracked using a gyroscope and accelerometer to ensure stability [2]. Mars Exploration Rovers utilize multiple localization methods, including the deployment of an accelerometer, gyroscope, magnetometer and motor encoders, to navigate an extraterrestrial environment [3]. To build a robust mobile robot upon which multiple missions could be based, a mechanism of acquiring precise localization information must be explored.

In the system architecture, a Host Computing Station (HCS) assigns global tasks to the mobile robot, and is operated either manually or autonomously. The robot can be operated in three modes: 1) navigated by a human operator, 2) navigated by an operator for only critical operation, and 3) autonomously navigated once given a destination. The mobile platform initially informs the HCS of its facing direction and receives destination coordinates information from the HCS. An embedded computer determines an optimal path and commands the platform to move between intermediate nodes that lie along the calculated path towards the intended destination. It uses the data acquired from the mobile platform's sensors to inform the HCS of the current platform location within a margin of error relative to the sensors. The sensor suite includes an accelerometer, a gyroscope, and a magnetometer. Unfortunately, inaccuracies in the output collected from each individual sensor introduce unreliable data to the system.

The sensors provide unique information about the current position of a robot. An accelerometer indicates the direction of gravity when at rest. A gyroscope indicates radial velocity. A magnetometer measures the strength of surrounding magnetic fields. The combination of these sensors constitutes an inertial measurement unit (IMU). Each of these sensors introduces noise to the system, and the gyroscope error accumulates, introducing drift. A magnetometer indicates direction, but results do not provide high precision measurements.

The paper focuses on implementation of direction cosine matrix (DCM) on a Cypress Programmable System on a Chip-5 (PSoC-5) platform to combine IMU sensor data and obtain accurate orientation information for a semi-autonomous robot under various terrain inclinations. This semi-autonomous robot is capable of navigating and mapping the surrounding environment. The DCM algorithm is demonstrated to obtain and maintain orientation accuracy under varied inclinations of pitch and roll.

The outline of the paper is organized as follows: Chapter 1 presents the background information and motivation for this project. Chapter 2 describes multiple processing units that comprise the system architecture of the proposed mobile robot. Chapter 3 discusses robot's hardware interface and the IMU drivers. Chapter 4 describes the direction cosine matrix algorithm implementation on PSoC-5. Chapter 5 compares the results for the orientation information between raw magnetometer sensor data and the DCM output data for various levels of surface inclinations. Chapter 6 concludes the paper and presents future work.

II. System Architecture

The robotic system architecture consists of two primary components, the HCS, and the onboard Embedded Hardware. The HCS provides visual environment maps to the robot operator. The HCS creates Level 1 and Level 2 maps from incoming aerial map data. Environment mapping is divided into three maps, defined at resolutions of 1km by 1km (Level 1), 100m by 100m (Level 2), and 10m by 10m (Level 3). Fig. 1 depicts the three-level map.

An operator specifies a starting point and ending destination on the Level 1 map. The HCS is responsible for calculating static optimal paths for Level 1 and Level 2 scaled maps and communicates the path information to the mobile platform. The embedded computer performs a dynamic path finding algorithm to traverse Level 3 maps [4].

The HCS also processes robotic sensor data, maps visualization data in an Open Graphics Library environment on three scaled levels, and tags multimedia data to the appropriate coordinate location. Finally, the HCS provides remote control capabilities in missions requiring human intervention. The system components are illustrated in Fig. 2.

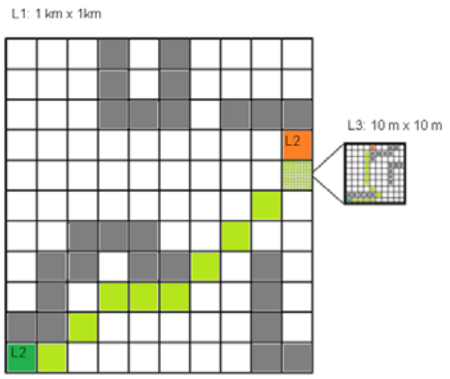

Fig. 1 Hybrid Routing Algorithm Model

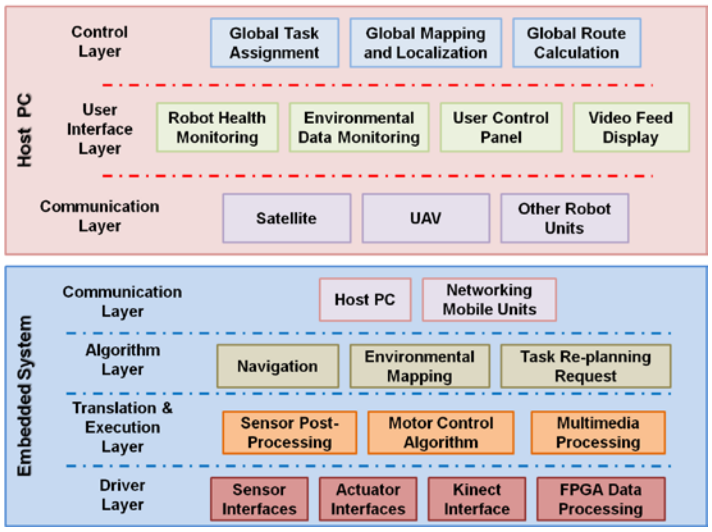

Fig. 2 System Architecture

The embedded hardware consists of three primary layers: the Algorithm Layer, Platform Layer, and Driver Layer. The Algorithm Layer is responsible for computing dynamic pathfinding algorithms, obstacle detection, and obstacle avoidance on Level 3 maps. The Platform Layer is responsible for the steering, sensor fusion, image processing, and kinematics of the mobile platform.

The Driver Layer contains the sensor and motor controller interfaces used for Robotic unit. The PSoC-5 serves as the microcontroller to support sensors, motor controllers, and communications for the mobile platform. The PSoC-5 provides the current localization of the mobile platform to the embedded computer through a serial port. The PSoC-5 performs data acquisition, applies control algorithms, and standardizes output, while the embedded computer handles all path-finding logic. An Analog-to-Digital Converter (ADC) samples data from range sensors. An Inter-Integrated Circuit (I2C) Master acquires data from the IMU. Pulse-Width Modulation (PWM) drives the motor controllers. One

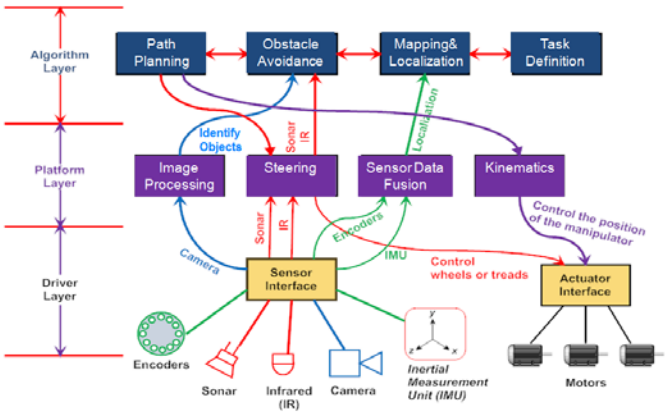

Fig. 3 System Layer Model

Universal Asynchronous Receiver/Transmitter (UART) connects the embedded computer or debugging terminal, and another connects the GPS. The system layer model is depicted in Fig. 3.

The PSoC-5 acquires kinematic sensor information. Navigation data is determined in the Algorithm Layer, passed through the Motor Control Algorithm block of the Platform Layer on the PSoC-5, and used to determine motor controller output. The kinematic sensors provide data for proper mapping and unit localization. IR and US sensors, along with Microsoft Kinect image acquisition, provide real time obstacle monitoring to allow the mobile platform to sustain autonomous and semi-autonomous missions. Fig. 4 illustrates the direction of information flow the embedded system layers, excluding communications.

Unique advantages of the Cypress PSoC-5 microcontroller include programmable voltage, instrumentation, and amplifiers. PSoC-5 allows the vendor to configure timers, counters, and PWM units, and each component is accessible through a proprietary application-programming interface (API). The PSoC-5 Digital Subsystem features configurable Universal Digital Block (UDB) units. Each UDB contains

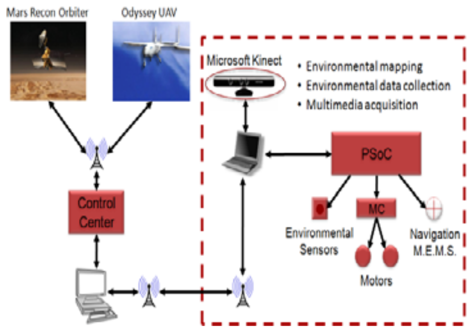

Fig. 4 System Layer Model

programmable array logic and programmable logic device functionality. A wide variety of peripherals are supported. In addition to the UDB array’s flexibility, PSoC-5 provides configurable dedicated digital blocks that target specific functions. The Analog Subsystem of PSoC-5 includes multiplexors, comparators, voltage references, op-amps, mixers, and Trans Impedance Amplifiers (TIA). They can be routed out from any GPIO pin [5].

UDB units utilized by the mobile platform include up to 32-bit PWM, counter, and timer UDB units. A Dedicated I2C communication blocks is also used. Multiple clocks can be configured to drive the digital components. A lack of parallel processing, small RAM capacity, and a maximum clock rate of 48MHz limit the PSoC-5. A controlling embedded computer is responsible for pathfinding computations.

The manufacturer of the PSoC-5 microcontroller publishes a free software suite called PSoC Creator to facilitate design implementation. This software allows drag-and-drop component placement. Basic components include wires and sheet connectors, while more complex components include I2C Master controllers and PWM blocks [6].

III. IMU DRIVERS

Several hardware components are supported by software to facilitate the DCM algorithm. IMU data acquisition across I2C transactions requires software to facilitate control over an I2C master, as well as communicate with the IMU. The I2C master addresses the sensors in the IMU independently, and output from the sensors is delivered in specific units. Multiple registers in the IMU contain sensor configuration data and measurement values.

A transaction to read the IMU is initiated by a start signal followed by a concatenated register address and a write signal sent from the I2C master. The master writes the address of the register that to be read to a pointer register in the IMU. The IMU sends an acknowledgement, and the master responds with a second start signal followed by the device address concatenated with a read signal. An acknowledgement of to begin reading is sent to the master. The IMU begins transmitting data from the register indicated by the pointer one byte at a time. A negative acknowledgement is sent when the transaction is concluded. This allows three-dimensional accelerometer, gyroscope, and magnetometer data to be six bytes long, two bytes for each dimension.

A series of vector functions are used store the output of the IMU. Vectors are stored in a structure containing three floating-point numbers. Cross product, dot product, vector addition, and normalization functions are required to determine the direction of gravity, acceleration, radial velocity, and magnetic heading.

Vectors store the x, y, and z elements of accelerometer, gyroscope and magnetometer sensor data. The raw output of the sensors is defined in units of least significant bits (LSB) [7]. Acceleration in g-forces is given as the product of the accelerometer value and the scaling factor 1g/16,384 LSB. For example, a sixteen bit value of 0010 0000 0000 0000 is equivalent to 8,192 LSB. The measured acceleration is given by:

$$8{,}192LSB \times 1g / (16{,}384LSB) = 1g \quad (1)$$

Gyroscope and magnetometer values are computed similarly. Gyroscope values are interpreted in units of degrees or radians per second per LSB, and magnetometer values are interpreted in units of gauss per LSB.

Functions to read the IMU are triggered by a timed interrupt, triggered when a timer expires. A control register is used to hold a reset signal on the timer high until the DCM has been updated.

Integration of the gyroscope utilizes another timer, which is restarted whenever the gyroscope has been read. This provides intervals accurate to 1 µs.

IV. The Direction Cosine Matrix

A DCM algorithm has been used to provide estimates of the robot localization to the PSoC-5 using accelerometer, gyroscope, and magnetometer data. The estimates are based on recursive DCM updates over small periods of time. The algorithm utilizes gyroscope values to track rotation of the inertial reference frame of the robot. A PI controller is used with the algorithm to reject external noise from the accelerometer and magnetometer, and correct gyroscope drift [8]. The PI controller also enhances compass heading results. The accelerometer, magnetometer, and gyroscope values provide input to the DCM algorithm.

It is necessary to maintain orientation information about the robot (body frame) with respect to the Earth (global frame). A global frame of reference (**V_G**) is found by the product of the rotation vector (**R**) and the body frame of reference (**V_B**), given an angle of rotation about each axis belonging to the body frame:

$$\overrightarrow{V_G} = \vec{R} \times \overrightarrow{V_B} \quad (2)$$

The rotations about principal axes (x, y, z) are defined in terms of radians as follows:

$$\overrightarrow{Rx} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix}$$

$$\overrightarrow{R_Y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\phi \end{bmatrix} \quad (3)$$

$$\overrightarrow{R_Z} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A complete rotation matrix is a product of three vectors:

$$\vec{R} = \overrightarrow{R_z(\varphi)} \times \overrightarrow{R_y(\theta)} \times \overrightarrow{R_z(\psi)} \quad (4)$$

The relation between the DCM and Euler angles are formed from the first column and last row of the resulting matrix [9]:

$$Pitch = -\arcsin(-\sin\phi)$$
$$Roll = -\arctan\left(\frac{\sin\phi}{\cos\phi}\right) \quad (5)$$
$$Yaw = -\arctan\left(\frac{\cos\psi}{\sin\psi}\right)$$

The DCM depends upon the rotation matrix, which transforms the orientation of the platform’s Euler Angles (roll, pitch, and yaw) to a reference coordinate system. The gyroscope measures angular displacement of the robot. The accelerometer and magnetometer are used to correct gyroscope drift. The error between the angular displacements indicated by the gyro must be checked against the direction of gravity, and the magnetic heading to correct drift.

The velocity of a rotating vector is defined as:

$$\frac{d\overrightarrow{r(t)}}{dt} = \overrightarrow{\omega(t)} \times \overrightarrow{r(t)} \quad (6)$$

where ω(t) is a rotation rate vector and r(t) is a tangential vector. If the initial conditions and the period between iterations are known, an approximation of the linear displacement of the vector is the cross product result of angular displacement dθ(τ) and rotational vector r(τ) over some time interval t.

The location of the vector at any time t is therefore given by:

$$\overrightarrow{r(t)} = \overrightarrow{r(0)} + \int_0^t d\overrightarrow{\theta(\tau)} \bullet \overrightarrow{r(\tau)} \quad (7)$$

The resulting equation updates the rotation matrix using incoming gyroscope information in terms of robot angular rotations is as follows:

$$d\theta_x = \omega_x \times dt$$
$$d\theta_y = \omega_y \times dt \quad (8)$$
$$d\theta_z = \omega_z \times dt$$

Equation 5.3.8 will eventually accumulate numerical errors attributed to rounding and gyro drift [10]. As error accumulates, the axes are reported as non-orthogonal and no longer represent a rigid body. Renormalization enforces the orthogonal relationship between the axes.

$$\vec{X} = \begin{matrix} r_{xx} \\ r_{xy} \\ r_{xz} \end{matrix}, \vec{Y} = \begin{matrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{matrix} \qquad (9)$$

$$\varepsilon = \vec{X} \bullet \vec{Y} = \vec{X}^T \vec{Y} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \end{bmatrix} \times \begin{bmatrix} r_{yx} \\ r_{yy} \\ r_{yz} \end{bmatrix} \qquad (10)$$

where ε is the error between the two vectors.

Apportioning half of the error for X and Y rows and rotating them in opposite direction by cross coupling leads to:

$$\vec{X}_{orthogonal} = \vec{X} - \varepsilon / 2 \times \vec{Y} \qquad (11)$$

$$\vec{Y}_{orthogonal} = \vec{Y} - \varepsilon / 2 \times \vec{X}$$

Next the Z row of the matrix is adjusted to be orthogonal to the X and Y vectors. Cross product of corrected X and Y leads to:

$$\vec{Z}_{orthogonal} = \vec{X}_{orthogonal} \times \vec{Y}_{orthogonal} \qquad (12)$$

Using Taylor's expansion to get magnitude adjustment equations for the row vectors yields:

$$\vec{X}_{normalized} = \tfrac{1}{2}(3 - \vec{X}_{orthogonal} \bullet \vec{X}_{orthogonal}) \bullet \vec{X}_{orthogonal} \qquad (13)$$

$$\vec{Y}_{normalized} = \tfrac{1}{2}(3 - \vec{Y}_{orthogonal} \bullet \vec{Y}_{orthogonal}) \bullet \vec{Y}_{orthogonal} \qquad (14)$$

$$\vec{Z}_{normalized} = \tfrac{1}{2}(3 - \vec{Z}_{orthogonal} \bullet \vec{Z}_{orthogonal}) \bullet \vec{Z}_{orthogonal} \qquad (15)$$

An accelerometer is used for roll-pitch drift correction is used in conjunction with a magnetometer. The roll-pitch rotational correction vector is computed by cross-multiplying 3rd row of DCM matrix with normalized gravity reference vector [11]:

$$\varepsilon' = \begin{bmatrix} r_{zx} \\ r_{zy} \\ r_{zz} \end{bmatrix} \times \vec{g}_{reference} \qquad (16)$$

Where ε' is a correction vector. Both of the correctional vectors are multiplied by their respective weights and fed to a Proportional Integral (PI) feedback controller. Then, the total correction is applied to the PI controller:

$$\omega p_{correction} = Kp \cdot \varepsilon' \qquad (17)$$

$$\omega i_{correction} = Ki \cdot \varepsilon'$$

The total correction vector is summed with the gyroscope information.

$$\overrightarrow{\omega(t)} = \overrightarrow{\omega(t)}_{correction} + \overrightarrow{\omega(t)}_{gyroscope} \qquad (18)$$

Finally, the DCM is updated utilizing the product of a time step and the corrected gyroscope vector:

$$\overrightarrow{R(t)} = dt \cdot \overrightarrow{\omega(t)}. \qquad (19)$$

The output of Euler Angles allows the mobile robot to report accurate rotation angles regardless of the inclination angle of a traversed plane. This maintains heading information when the robot encounters sloped or rough terrain. Figure 5 represents the DCM algorithm, as implemented on the PSoC.

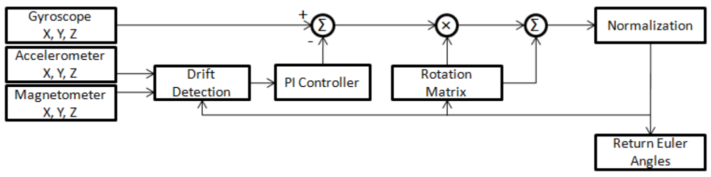

Fig. 5 System Layer Model

V. Software Implementation of The DCM

The DCM implementation adapts existing Arduino software, and the uses custom IMU sensor and math functions [12]. The source software was written in C++, which the PSoC-5 does not support. The source software also implemented a different sensor suite than the one utilized by the robot.

The original software implemented matrices as objects that were defined as two-dimensional arrays of any size. A constructor would be called to define the number of rows and columns of a matrix. The equality, addition, subtraction, and multiplication operators were overloaded to perform unique matrix operations, each returning a newly created matrix object. In C, neither can objects be declared, nor operators overloaded. Each matrix is defined as a three-by-three array of floating-point numbers. Functions were created to support matrix operations, which take three matrices, passed by value, as input. The first two matrices are the operands, and the third stores the result of the operation.

The DCM is stored into a three-by-three floating-point array, representing. The roll, pitch, and yaw are respectively stored along the diagonal. The DCM is initialized to zero, and the accelerometer and magnetometer are read to provide the error estimates to the PI controller. The DCM does uses the accelerometer compare the roll and pitch measured by the accelerometer against those obtained by the DCM. This is implemented in a function to correct drift in the gyro.

$$\begin{aligned} MAG\_X = {} & m.x \cdot \cos(pitch) \\ & + m.y \cdot \sin(roll)\sin(pitch) \\ & + m.z \cdot \cos(roll)\sin(pitch) \end{aligned} \qquad (20)$$

$$MAG\_Y = m.y \cdot \cos(roll) - m.z \cdot \sin(roll) \qquad (21)$$

where m is the magnetometer reading, and MAG_X and MAG_Y are the tilt compensated components of the heading. The previous heading recorded in the DCM is stored in the first row of the matrix. The difference between these values and the tilt compensated magnetometer values provides an estimate of the error present in the DCM. The PI controller uses the error to offset the measured yaw. The error of the pitch and roll is determined by the cross product between that recorded in the DCM, and that measured by the accelerometer, as given in Eq. 5.4.15. This result is relative to the magnitude of the error between the two vectors. The PI controller scales the error, and adds to the pitch and roll recorded in the DCM.

The gyroscope is integrated over time. The result of the arc measured by the gyroscope and the PI controller are added, and used to update the DCM. The matrix is then normalized, and Euler angles are computed.

VI. Experimental Results and Discussions

An apparatus built to allow three axis rotations was used to test the navigation capability of the robot and to verify localization algorithms utilizing the IMU. This testing apparatus allows independent motion along each of three axes: roll, pitch, and yaw. Three methods of angular measurement were observed: 1) The heading provided by raw magnetometer data, 2) Raw, integrated, gyroscope data, and 3) Direction Cosine Matrix data utilizing a PI controller and the IMU. Each test case examines yaw information under fixed pitch positions. The IMU is affixed to a plate of wood on top of the apparatus. Wood was chosen because it does not influence magnetometer in the IMU.

To test the varied angular data acquisition methods, the servo that controls yaw is calibrated. The duty cycle to a PWM driving the pitch servo is experimentally investigated until the values that bring the servo nearest to 90 degrees from an initial position are found. Then the values that bring the pitch servo nearest 20 degrees pitch are determined. The compass heading determined by the DCM algorithm is observed once the servo is determined to be optimally configured. Values provided by the unfiltered gyroscope, and magnetometer, are also observed during the experiment.

The initial angle is reported by the gyroscope. For raw gyroscope values, this is zero degrees. The angular rate determined after servo is rotated to 90 degrees is reported when the servo stops moving. A short delay between executing servo motion allows the two filters to stabilize before initializing the next rotation. This delay is retained when measuring the raw gyroscope values to maintain identical test conditions between each experiment. Results are analyzed in terms of mean angle traversed in the clock-wise and counter-clockwise direction, and the mean heading in each direction. The mean angles and standard deviations are given for each experiment.

The servo calibration found the angle of rotation nearest 90 degrees to be 90 ± 2 degrees. The pitch angles of the platform were determined to be zero, pitched down 20 degrees, and pitched up 20 degrees. The initial heading is approximately 45 degrees east of geographic north. At each inclination, the yaw servo was rotated clockwise 100 times, and counter-clockwise 100 times. The resulting measure at the end of each rotation was tabulated. When using the DCM and the magnetometer, the measure reported is the compass heading. When using gyroscope values, the measure reported is relative to the initial yaw of the sensor. The result data is tabulated and includes the counter-clockwise measurement, clockwise measurement, and the difference between each measurement. Table 1 demonstrates the results.

Table 1. DCM, Gyroscope, and Magnetometer Results

Pitch:	Mag	°CW	°CCW	d□
0°	μ	-49.86	51.45	101.33
	σ	2.84	3.44	4.26
10°	μ	-51.53	52.04	103.57
	σ	2.12	3.16	3.63
20°	μ	-50.46	47.08	97.54
	σ	2.67	3.45	4.37
30°	μ	-45.69	42.63	88.32
	σ	2.55	2.85	3.71
40°	μ	-45.25	35.51	80.76
	σ	2.39	2.28	3.5
	Gyro	°CW	°CCW	d□
0°	μ	0.26	89.2	88.95
	σ	0.54	0.45	0.56
10°	μ	0.7	89.92	89.22
	σ	1.13	1.03	1.69
20°	μ	0.28	86.84	86.57
	σ	0.82	0.73	1.03
30°	μ	0.4	79.17	78.77
	σ	0.62	0.36	0.68
40°	μ	-0.19	70.82	71.01
	σ	0.84	0.74	1.17
	DCM	°CW	°CCW	d□
0°	μ	-42.8	47.71	90.51
	σ	0.92	0.92	0.45
10°	μ	-44.66	47.77	92.43
	σ	1.4	1.27	0.86
20°	μ	-42.16	48.59	90.74
	σ	5.61	6.15	4.93
30°	μ	-42.8	47.42	90.23
	σ	3.2	1.52	3.23
40°	μ	-39.71	49.86	89.57
	σ	4.28	2.26	5.82

Pitch: Angle of Inclination of the IMU
μ: Mean
σ: Standard Deviation
°CW: Heading Measured at Furthest Clockwise Position
°CCW: Heading Measured at Furthest Counter Clockwise Position
d□: Total Arc Traversed

The DCM meets or exceeds the accuracy of the magnetometer at all inclinations of pitch, especially when used to measure the traversed arc. The standard deviation of the DCM demonstrates improved heading information at low angles of inclination. At steeper pitches, the standard deviation of the DCM is near that of the magnetometer, though the mean accuracy of the angles and arc measured is improved by the DCM. The gyroscope provides a lower standard deviation at all inclinations, though the accuracy is greatly impacted at all inclinations beyond 10° pitch. Figure 6 depicts the testing apparatus.

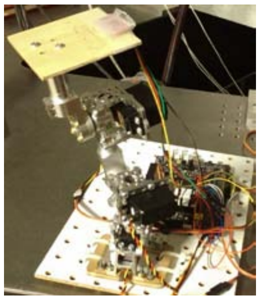
Fig. 6 Testing Apparatus

VII. Conclusion

A semi-autonomous robot requires control algorithms to accurately provide localization data. The PSoC-5 provides sufficient computational power to process algorithms to localize a mobile robot. A direction cosine matrix provides unique advantages when applied to localization.

Experimental results demonstrate the implemented DCM provides accurate measurements and improves the reliability of the magnetometer at various angles of inclination.

Future work includes replacing the PI controller in the DCM with a Kalman Filter and implementing GPS in the Kalman Filter to augment heading information of the robot as it changes position. These improvements would enhance the DCM control algorithm, and localization of a mobile robot.

References

[1] Kriechbaum, K.L. 2006. "Tools and Algorithms for Mobile Robot Navigation with Uncertain Localization." PhD diss., California Institute of Technology. <http://thesis.library.caltech.edu/2363/1/kriechbaum-thesis.pdf>.

[2] Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M. and Kawatsuma, S. 2013. "Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots." Journal of Field Robotics, 30: 44–63. DOI: 10.1002/rob.21439

[3] Powell, M.W., Crockett, T., Fox, J.M., Joswig, J.C., Norris, J.S., Rabe, K.J., McCurdy, M., and Pyrzak, G. 2006. "Targeting and Localization for Mars Rover Operations." Paper presented at the IEEE International Conference on Information Reuse and Integration, Waikoloa Village, Hawaii, September 16-18. DOI: 10.1109/IRI.2006.252382.

[4] Stentz, A. 1996. "Map-Based Strategies for Robot Navigation in Unknown Environments." Paper presented at the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems, March. http://www.ri.cmu.edu/pub_files/pub2/stentz_anthony__tony__1996_1/stentz_anthony__tony__1996_1.pdf

[5] .PSoC ® 5: CY8C55 Family Datasheet URL: http://ecologylab.net/courses/sensoryInterfaces/resources/CY8C55DataSheet.pdf

[6] PSoC ® Creator ™ Quick Start Guide URL: http://www.cypress.com/index.cfm?docID=45972

[7] InvenSense, Inc. 2012. "MPU-9150 Register Map and Description Revision 4.2." Sunnyvale. http://www.invensense.com/mems/gyro/documents/RM-MPU-9150A-00v4_2.pdf>.

[8] D. Choukroun, H., Weiss,Bar-Itzhack, I.Y., Oshman, Y. 2010. "Direction cosine matrix Estimation from Vector Observations using a Matrix Kalman Filter." IEEE Transactions on Aerospace and Electronic Systems 46, 1: 128-136. DOI: 10.1109/TAES.2010.5417148

[9] Premerlani, W. 2010. "Computing Euler Angles from Direction Cosines." URL: <https://gentlenav.googlecode.com/files/EulerAngles.pdf>.

[10] Premerlani, W. and Bizard, P. 2009. "Direction Cosine Matrix IMU: Theory." URL: <http://gentlenav.googlecode.com/files/DCMDraft2.pdf>.

[11] Macias, E., Torres, D., Ravindran, S. 2012. "Nine-Axis Sensor Fusion Using the Direction cosine matrix Algorithm on the MSP430F5xx Family." Texas Instruments Application Report, Dallas. <http://www.ti.com/lit/an/slaa518a/slaa518a.pdf>.

[12] Pololu Corporation, "MiniIMU-9-Arduino-AHRS," URL: <https://github.com/pololu/minimu-9-ahrs-arduino>