International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 26115-26126

http://www.ripublication.com

Human Detecting and Location Mapping Robot Using PIR Sensor and Velocity Motion Model

[1]Reethu Mariam Raphael, [2]Sreerag Veppil Sasikumar, [3]Vaishnav Mohanan Mavilakandy,[4]Vishnu Sudheer Menon, [5]Praveen Kumar
[1]Student- Department of EEE, Amrita school of engineering, Coimbatore
reethumariam36@gmail.com
[2]Student-Department of EEE, Amrita school of engineering, Coimbatore
mail2sreerag@gmail.com
[3]Student-Department of EEE, Amrita school of engineering, Coimbatore
vaishnavmohanan.m@gmail.com
[4]Student-Department of EEE, Amrita school of engineering, Coimbatore
vishnu_291093@hotmail.com
[5]Asst. professor-Department of EEE, Amrita School of Engineering, Coimbatore
praveen_mtech82@yahoo.co.in

Abstract

This paper proposes a novel method to rescue humans trapped after an earthquake. It revolves around constructing an autonomous robot that can navigate and map the destroyed terrain and also locate any humans that are alive in that terrain. The navigation of the robot is done with the help of the on-board ultra-sonic sensor and is localized using Velocity Motion Model [1]. The trapped humans are located through an on-board PIR sensor. The ultra-sonic sensors and the PIR sensors are connected to the Arduino UNO board and are interfaced with a Raspberry Pi b+[2] micro controller through serial USBcommunication. The algorithm for navigation of the robot and mapping the unknown terrain is done using Velocity Motion Model, which will be run in the Raspberry Pi after the robot has finished moving through the terrain. On the whole, this paper illustrates how robotics can be used for humanitarian purposes and also provides a cheap alternative to the traditional means of human body detection.

Keywords: Velocity Motion model, Raspberry pi, Artificial intelligence, Robotics, Humanitarian, Disaster Management

Introduction

Calamities can never be predicted it can only be acted upon once struck. The important factor that determines the number of survivors is the quality of the rescue operation. But the problem at present is the lack of a novel method in locating living beings trapped under the debris within the shortest duration of time. The existing rescue methods usually involve using sniffer dogs, acoustic sensors, man labour etc. However, these methods are highly time consuming and mostly lead to poor outcome. With the advancements inscience and technology, we plan to tackle these problems by introducing an efficient system that can be used in the fields of disaster management, search and rescue.

We are implementing a land based mapping system [4][5][6] which has several advantages and it outweighs the existing methods. This can be implemented in a larger scale by increasing the number of robots through efficient algorithms. Unlike manual labour which requires time for organising and coordinating a rescue effort, this method would easily be able to isolate and locate the trapped beings. This system consists of a robot controlled by a Raspberry Pi B+ controller, Arduino with peripherals such as Ultra-Sonic Sensors and PIR Sensor. The robot is run on 4 wheels by using a motor on each wheel and powered by battery. As a result the robot will be able to run for 30 minutes continuously. The prime objective of the robot is to map the new terrain and locate any humans that are alive in the terrain. The mapping of the robot is done by using the Velocity Motion Model and the detection of the humans is done by using PIR sensors. The output obtained from the micro controller is set to coordinates that correspond to the location of the landmarks and humans. These coordinates are then mapped out and position of the trapped victims can be easily estimated. This method is comparatively cost effective, thus making it a feasible method to be used during disasters like earthquakes.

Proposed System

This system consists of a 4-wheeled robot fitted with 4 ultrasonic sensors on each of its side and a PIR sensor on the front side. These sensors are connected to an Arduino-UNO board and this board is connected to a Raspberry Pi microcontroller board. The raspberry and Arduino -UNO is powered by a 5V 1500 battery bank. The communication between Arduino-UNO and Raspberry Pi is achieved through a serial USB connection between these 2 micro controllers. The data from the Arduino is read by the raspberry Pi by using the pyserial library present in Python. The motors are powered by a 12 V battery and it is controlled by the raspberry pi using two L-293D ICs.

The navigation of the robot is done with the help of the Ultra-Sonic Sensors. The range finding sensors will warn the robot of any obstacles in its path. So if the robot encounters an obstacle in its forward motion, it is programmed to check the left of its position for any available path. But if the left of its position is also blocked then the robot will check its right or it will move back if the right is also blocked. Moreover if the robot is placed in a long stretch of land, then the robot is programmed in a way to move in zigzag motion in order to cover the entire stretch of land.

During the motion of the robot, it will update its position by using the velocity motion model. Every 2 seconds of its forward motion it will update its co-ordinates and on taking a turn this function will modify its position accordingly. So finally we will be getting a list consisting the position of the robot at regular intervals. These positions are mapped using a Matplotlib library in Python.

The PIR Sensor is used here to detect the presence of human beings in the path of the robot. This sensor will alert the raspberry pi of the presence of a human and by using the ultra-sonic sensor we will be getting the distance of the human from the robot. So with the distance the approximate position of the robot will be calculated by the velocity motion model and it will be plotted in the map.

Mathematical Modeling

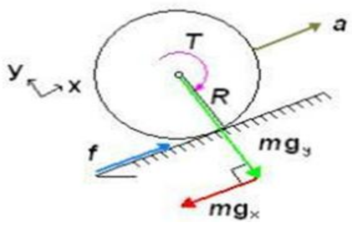

Figure 1: Free body diagram of the Robot wheel

The robot will move in three ways: up the inclined slope, along the horizontal plane and down the inclined plane. Considering the torque to be maximum when it moves up the inclined plane, the calculations are done. In order to move on a horizontal surface, a wheeled robot's motors must produce enough torque to overcome any imperfections in the surface or wheels, as well as friction in the motor itself. Therefore theoretically, a robot (small or large) does not require much torque to move purely horizontally. The torque required to move up an inclined plane is more than the torque required to move along a horizontal plane.

For a robot to move up an incline at a constant velocity with no acceleration or deceleration, it must produce enough torque to overcome the effect of gravity, which would otherwise cause it to roll down the incline. On an inclined surface i.e. at an angle theta (θ), only one component of its weight, mg_x parallel to the surface, causes the robot to move downwards. The other component, mg_y is balanced by the normal force the surface exerts on the wheels. There must be friction between the wheel and the surface so that the robot does not slide down the incline. The components of weight are given by:

$$mg_x = M*g\sin\theta \tag{1}$$

$$mg_y = M*g\cos\theta \tag{2}$$

We have considered the worst case scenario for select the specification of the motor that is when the robot is moving up an inclined plane with acceleration (a).

In the above free body diagram the component of weight mg_x is opposed by the friction (f) acting between the surface and the wheels and the component mg_y is balanced by the normal force the surface exerts on the wheels.

Torque is given by: T=f*R (3)

where R is the radius of the wheel.
Balancing the forces in the x direction

$$\sum F_x = M * a = f - M * g_x \quad (4)$$

$$M*a = T/R - M*g\sin\theta \quad (5)$$

$$T=R*M*a + R*M*g\sin\theta \quad (6)$$

Total torque required to accelerate the robot up an inclination is given by:

$$T=R*M*(a+g\sin\theta) \quad (7)$$

Where M is the total mass acting on the wheel
Torque required for each drive motor is given by

$$T = \frac{R*M*(a+g\sin\theta)}{n} \quad (8)$$

where n is the total number of drive wheels.
This torque is multiplied by a factor 1.5 to overcome the inefficiencies.

Power is given by P=T*ω (9)

whereω is the angular velocity.

$$\omega = V/R \quad (10)$$

where V is the velocity

$$\omega = \frac{(2*\pi*N)}{60} \quad (11)$$

where N is the speed of the motor.

Maximum current is given by, $I=\frac{P}{v}$ (12)

where v is the supply voltage

Capacity of the battery pack is given by C=I*t (13)

where t is the time of operation of the robot.
Total battery requirement = battery required per motor * number of drive motors (14)

Calculation
From equation 8

$$T = \frac{R * M * (a + g\sin\theta)}{n}$$

R=3.5cm
M=mass of chassis +mass of 4 motors + mass of Raspberry pi b^+ microcontroller + mass of Arduino + +mass of battery + mass of sensors
=250+150*4+200+200+500+250 =2000gm =2Kg
V=0.5m/s
t=100ms

$a = V/t = 0.5*1000/100 = 5 m/s^2$
$g = 9.8 m/s^2$
$\theta = 60$ (max inclination possible)
$n=4$
Time of operation of robot =1 hr.

$$T = \frac{0.035 * 2 * (5 + 9.8 * \sin 60)}{4} = 0.236\text{Nm}$$

T= 1.5*0.236 =0.35Nm
From equation 10 ---- ω= V/R = 0.5/0.035 = 14.285 rad/s
From equation 11 ---- N = $\frac{(60*14.285)}{2*\pi}$ = 136.42 rpm
From equation 9-----P= T* ω = 0.35*14.285=5.06W$\approx 6W$
From equation 12----I=P/v = 6/15 = 0.4A+0.7(for microcontroller) =1.1A
From equation 13----C=I*t=1.1*1=1.1Ahr
From equation 14----Total battery capacity =1.1*4=4.4Ahr$\approx 5Ahr$

Supply Circuit

The supply circuit consists of a battery bank of rating 5V, 1A which powers the motor driver L293D, the Raspberry Pi b+ microcontroller of rating 5V, 1A. This Raspberry Pi b+ micro controller powers the Arduino micro controller of rating 5v, 0.6A.

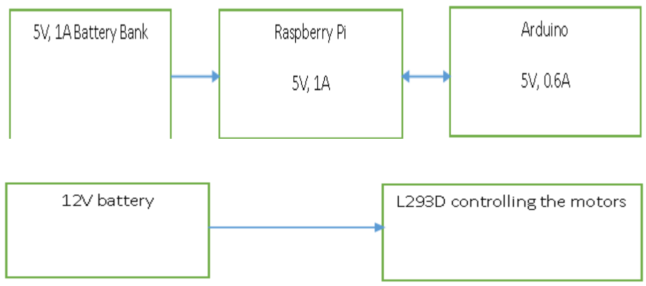

Figure 2: Supply Circuit

Working

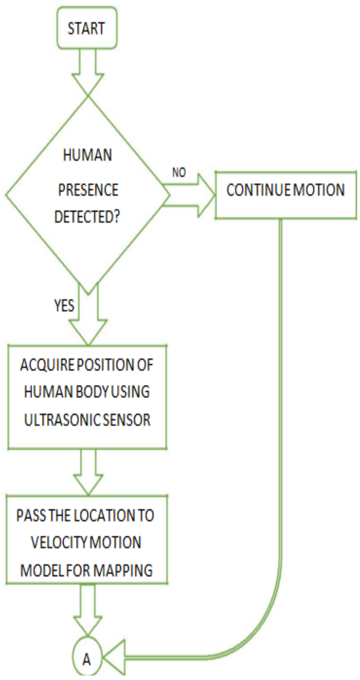

Figure 3(a): Flowchart

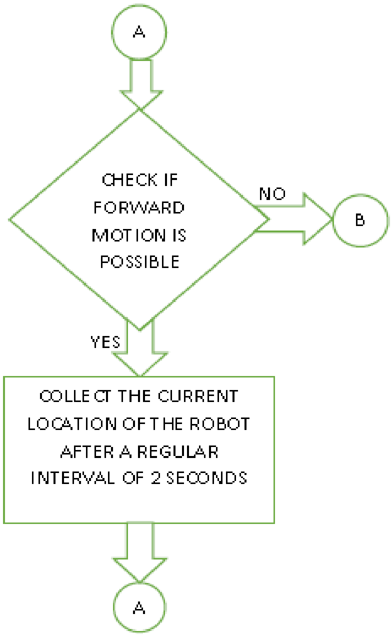

Figure 3(b): Flowchart

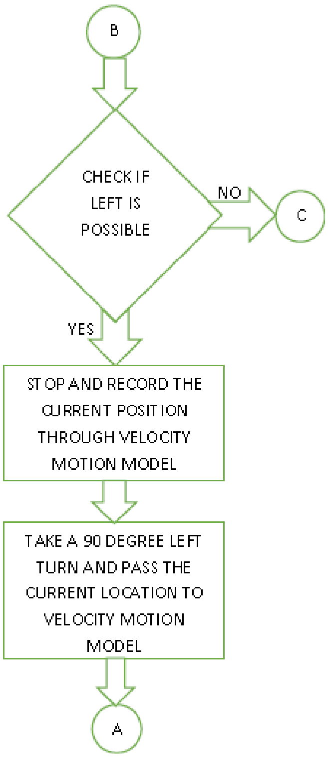

Figure 3(c): Flowchart

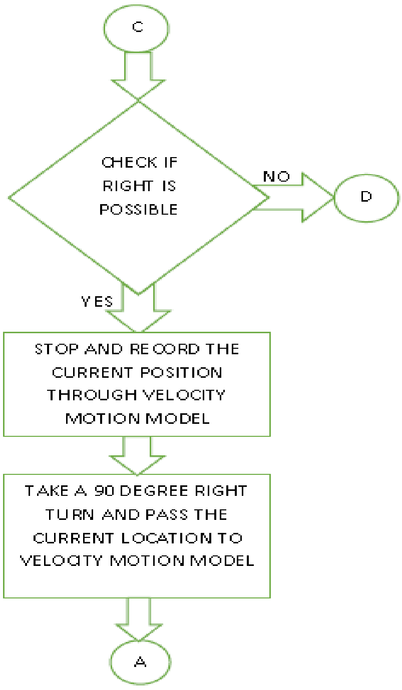

Figure 3(d): Flowchart

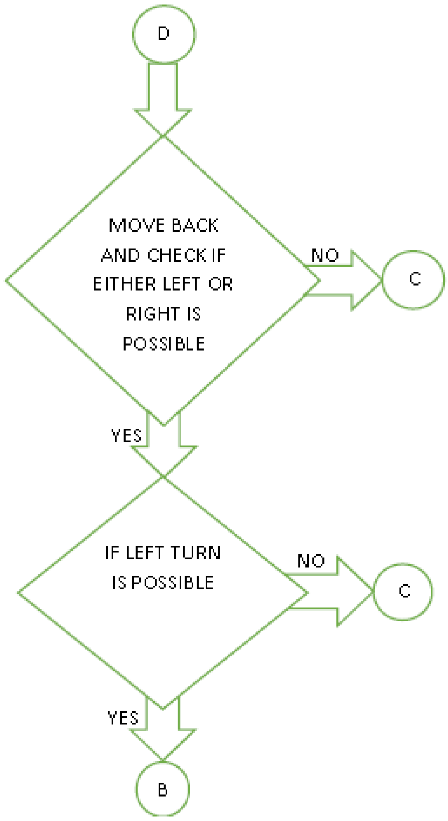

Figure 3(e): Flowchart

The flowchart (fig.3) shows the working of the autonomous robot.

Velocity Motion Model

Velocity motion model[1] assumes that the motion data u_t specifies the velocity commands given to the robot motors. Here p $(x_t \mid u_t, x_{t-1})$ gives the state transition probability and x_t and x_{t-1} are both robot poses (and not just its x-coordinates), and u_t is a motion command. The robot's pose comprises of its two-dimensional planar coordinates relative to an external coordinate frame, along with its angular orientation. Denoting the former as x and y (not to be confused with the state variable x_t), and the latter by θ, thus the pose of the robot is described by the following vector: $\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$. The velocity motion model assumes that a robot can be controlled through two velocities, a rotational and a translational velocity. The translational velocity at time t is given by v_t, and the rotational velocity is given by ω_t. We assume that positive rotational velocities ω_t induce a counter clockwise rotation (left turns) and positive translational velocities v_t correspond to forward motion. Hence, the motion command is given by

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \quad (15)$$

It is assumed that the inputs are initial pose $x_{t-1} = (x\ y\ \theta)^T$, a control command $u_t = (v\ \omega)^T$ and a hypothesized successor pose $x_t = (x`\ y`\ \theta`)^T$. It outputs the probability $p(x_t \mid u_t, x_{t-1})$ of being at x_t after executing control u_t beginning in state x_{t-1}, assuming that the control is carried out for the fixed duration Δt. The parameters α_1 to α_6 are robot-specific motion error parameters. $\alpha_1$and α_2 corresponds to translational error, α_3 and α_4 corresponds to angular error, α_5 and α_6 corresponds to error due to rotation noise. v_cap, w_cap, r_cap,corresponds to the actual translational velocity, rotational velocity and the inclination with the errors considered.

Algorithm

Algorithm velocity_motion_model(u_t,x_{t-1})

1.v_cap = v + sample ($\alpha_{1|v|} + \alpha_{2|w|}$)
2.w_cap = w + sample ($\alpha_{3|v|} + \alpha_{4|w|}$)
3.r_cap = sample ($\alpha_{5|v|} + \alpha_{6|w|}$)
4. z = v_cap /w_cap
5. $x' = x - z \sin\theta + z \sin(\theta + (w\_cap)\Delta t)$
6.$y' = y + z \cos\theta - z \cos(\theta + (w\_cap)\Delta t)$
7.$\theta' = \theta + (w\_cap)\Delta t + (r\_cap)\Delta t$
8. return $x_t = (x'\ y'\ \theta')^T$
Sample_normal distribution
1.k= b/6
2.return $(k\sum_{i=1}^{12} rand(-1,1))$

Thus the robot moves according to the flowchart shown in fig.3 and the new position of the robot x_t is determined using Velocity Motion Model algorithm. The positions thus obtained are mapped to find the location of the alive human trapped in an unknown terrain.

Hardware Implementation

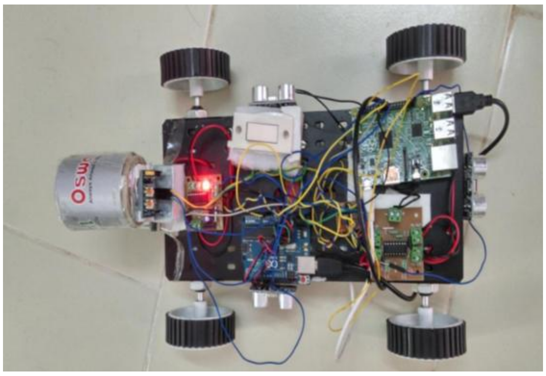

Figure 4: Top view of the Robot

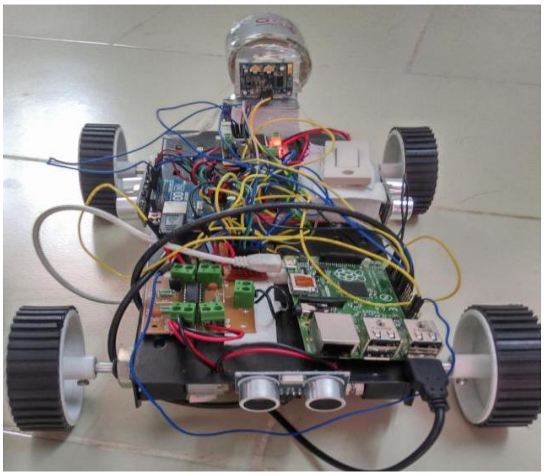

Figure 5: Rear view of the Robot

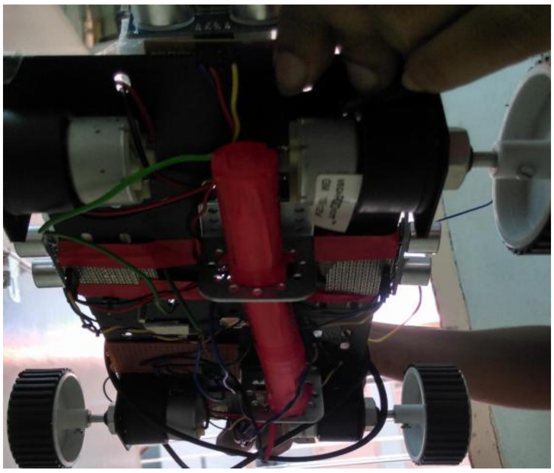

Figure 6: Bottom view of the Robot

The above fig.4, fig.5, fig.6 shows the hardware implementation of the autonomous robot, with 4 ultrasonic sensors on all the four sides and a PIR sensor in the front. Raspberry pi b+ micro controller, Arduino microcontroller, 2 motor drivers (L293D) are mounted on the top of the robot. The battery bank and 12V battery are attached to the bottom of the robot.

Experimental Setup

Figure 7: Experimental setup

An experiment was performed in the corridor with three people standing at random places. The goal was to map the trajectory of the robots motion and pin point the location of the human beings.

Experimental Results

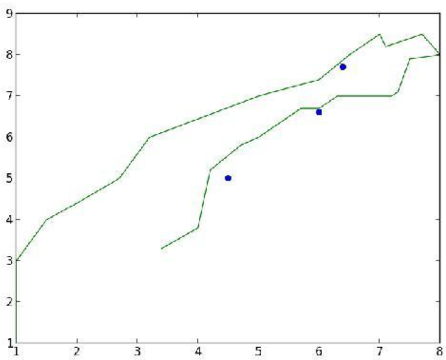

Figure 8: Experimental Result

The above figure shows the result of the experimental setup that is mapping of the robot motion and detection of human beings. In fig.8, the path of the robot is mapped and the location of the humans is indicated by dots.

Conclusion

In this 21st century, we need not depend on the traditional methods of disaster management. With the advancements in the field of robotics, it provides us an

efficient alternative replacing the age old means of finding or detecting human bodies trapped under any debris. Unlike humans, robots don't have the limitations of going into an unknown terrain.
Therefore, this method can be used for detecting trapped humans and thereby the quality of rescue operation can be improved.

References

[1] S. Thrun, W. Burgard and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.
[2] M. Richardson and S. Wallace, *Getting started with Raspberry Pi*. Sebastopol, CA: O'Reilly Media, 2012.
[3] W. Donat, *Learn Raspberry Pi programming with Python*. .
[4] S. Riisgaard and M. R. Blas, *SLAM for dummies: A tutorial approach to simultaneous localization and mapping.*
[5] S. Thrun, 'The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures', *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403-429, 2006.
[6] J. Mueller, *Beginning programming with Python for dummies*. Hoboken, New Jersey: Wiley, 2014.