

# Using a Nintendo Wii Remote to Help Navigate a Robot

Luke R. Sathrum, *Master's Student, California State University, Chico*

**Abstract** – This paper presents an idea of using infrared technology to help a robot locate itself in its environment. Through the use of infrared transmitters and a Nintendo Wii Remote a robot will be able to pick up different infrared signals and navigate using those signals. Topics covered include an overview of different robot localization methods, the Nintendo Wii Remote Technology, and an algorithm to use Nintendo Wii Remote Technology to help a robot navigate itself. Improvements to the process as well as future research are also discussed.

## I. INTRODUCTION

The use of robotics systems have been increasing year after year as more systems become automated and streamlined. While these systems have been progressing so has the technology to make these robots perceive their surroundings. Both Location Aware Computing and Robot Localization have become fields for researchers to try and develop methodologies and algorithms to best track where a robot or computer may be relative to its environment. Researchers have looked at many different technologies and methodologies in having a robot locate itself in its surroundings in an effort to create Robot Localization. These methods stem from mimicking the way humans and animals locate themselves and how they integrate their senses to achieve this. In the subsequent sections we will look at the three main ways that robots mimic localization and then describe the approach that will be used in our system of using a Wii Remote to help localize a robot.

### A. Visual

For humans it is very easy for us to use our eyes to tell us where we are located in a room, a field, or anywhere else. Our eyes and brain do a great job of communicating together to help

perceive our location. For computers and robots it is not as easy. A computer must accurately track the path of 3D trajectories using image-based sensors or cameras [1]. In a study for using path tracking with laser welding robots de Graaf, et al [1] are able to locate the robotic arm and where it is welding with accuracy no less than 0.1mm accuracy [1]. While their method is accurate it requires a lot of overhead processing to convert a 3-D image into something that a robot could understand. It is also very costly [1]. Another visual idea, which is the emphasis of this article, is to use line of sight infrared (IR) to help a robot locate where it is. If a robot is receiving an IR line of sight signal then it knows that the path between itself and the IR signal is clear.

### *B. Tactile*

Another way that we interact with our environment in helping locate where we are is through our tactile sense. This is emulated in robots through using proprioceptive sensors as seen in the experiment by Salter et al. [2]. While proprioceptive sensors are used for human-robot interaction they can also possibly be adapted to help with making a robot location aware. Humans use a combination of senses to determine their location including touch. Your skin can detect when you are in a cold room, when the wind is blowing at you, or when you are up against a wall. In the same way a robot can be fitted with proprioceptive sensors that can help it determine these things. While probably not a good main method for robot localization it can be used in conjunction with other methods to get a better picture of where a robot is located in relation to its surroundings.

### *C. Hearing*

While the idea of echo location is interesting in relation to robotic localization we will instead look at hearing in relation to telling how close we are to the sounds around us. Humans can hear

sounds from multiple locations and figure out which sound is the loudest. If all the sounds are sent at the same volume then it is easy to tell which sound is the closest in relation to where someone is. This idea has been adapted to help with computer localization in a paper by Ladd et al. [3]. Ladd uses an Ethernet 802.11 wireless signal to replace the audio signal that one might hear and uses a wireless card to receive signal strengths from multiple wireless base stations [3]. By using the signal strengths in conjunction with a receiver they are able to determine where a computer is located in a room.

#### *D. The Approach*

After looking at the different methodologies and also looking at the technology available it was determined that a method using line of sight infrared could be used to help a robot locate itself in a controlled environment. The approach I took was to adapt this idea to help create locations that an IR equipped robot could enter and easily navigate. While taking in cost considerations as well as the scope of the project it was determined that a Nintendo Wii Remote [4], which includes built into it an infrared camera and Bluetooth capability, could be used, in conjunction with IR transmitters, to locate itself and navigate through a maze. Beyond having an IR camera and Bluetooth the Wii Remote is also equipped with an accelerometer which could possibly be integrated with the IR technology, using Dead Reckoning [3], to help a robot locate itself when it has lost connection to all IR transmitters. The following sections will detail how my idea is implemented and the algorithm needed to complete the methodology. In Section 2 I will deal with the technologies needed to implement this idea by looking at both the IR transmitters and the Wii Remote. In Section 3 I will look at how the experiment is set up and also look at the algorithm needed to run the experiment. In Section 4 I present the experiment results. After analyzing the experiment I will give my ideas for improvement in Section 5. In Section 6 I look

at future research and in Section 7 I will wrap up my paper with my conclusion.

## II. TECHNOLOGIES

In order to implement my methodology we must first look at the technologies I will be using to achieve our goal. The main technologies that I used were the Wii Remote, which I use for its IR camera and ability to interface through Bluetooth to a computer, some infrared transmitters, Brian Peek's .NET Managed Library for Nintendo's Wiimote [5], a .NET library that can interface with a Wii Remote, and a radio controlled car, nicknamed the 'Lukii', that I will mount a Wii Remote onto.

### *A. The Wii Remote*

Because Nintendo has not released detailed design specifications for the Wii Remote, due to the fact that it is used with its currently popular Wii Game System, most of the following information has been reversed engineered. The Wii Remote is made up of many technologies that are integrated into a single wand like controller. The Wii Remote includes Bluetooth wireless technology, an accelerometer, an IR camera, eleven buttons, a rumble feature, an integrated speaker, and an extension port [6]. Since most of these technologies will not be used in the experiment we will not go into any detail about them.

#### *A.1 Bluetooth and the Wii Remote*

The reason that we are able to use the Wii Remote in our experiment is due to the fact that Nintendo created its Wii Remote using the well known Bluetooth standard [6]. This standard allows for a Wii Remote not only to be connected to a Wii Gaming System but also to any computer or device with a Bluetooth adapter. While the BlueSoleil Bluetooth stack is recommend to use with a Wii Remote and Windows [7] the default Windows drivers will work. The Wii Remote Interfaces with a host computer by sending hexadecimal reports over the

Bluetooth connection. These reports can be used with programming libraries, such as ‘Brian Peek’s .NET Managed Library for Nintendo's Wiimote’ to access the Wii Remote’s features.

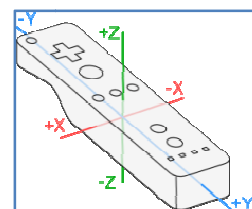
### *A.2 IR Camera and the Wii Remote*

The Wii Remote includes an IR Camera that can capture and track IR sources. This IR camera is capable of returning sets of data reports over Bluetooth. The Wii Remote has a built in processor that is capable of tracking up to 4 moving objects [6]. These 4 objects are assigned to ‘slots’ on the Wii Remote, the first IR source detected defaulting to slot one, and other sources being assigned to the next 3 slots available respectively. If an object moves out of view, its slot is marked as empty, but other objects retain their slots [6]. This means if slot 1 and 2 are allocated and the Wii Remote loses connection with the IR source for slot 1 then slot 1 becomes empty and slot 2 retains its IR source. In my experiment we will only be using slot 1, but multiple slots allows for expandability for the future.

### *A.3 Accelerometer and the Wii Remote*

The Wii Remote incorporates a three-axis accelerometer that is able to measure acceleration in three directions. It uses an Analog Devices ADXL330 integrated circuit to measure accelerations over a range of at least +/- 3g with 10% sensitivity [8]. Bandwidths for the integrated circuit can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis [9]. Because the accelerometer is affected by gravity it reports acceleration on its Z axis when lying still due to the fact that gravity is still pulling on it at a rate of about  $8.8 \text{ m/s}^2$  [6]. This means in free fall the accelerometer reports no acceleration on its Z axis. It is possible, through software, to calibrate the Wii Remote so that it will report no acceleration when it is

**Fig. 1: Wii Remote**

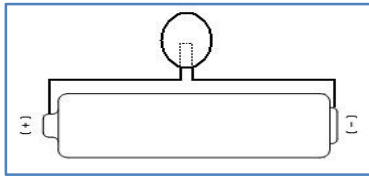


not moving [6]. Figure 1 [6] shows the 3 axis in relation to the Wii Remote

### *B. The Infrared Transmitters*

For my infrared transmitters I use an IR LED and a AAA battery connected together to produce an IR signal that a Wii Remote Camera can capture. The IR LED Emitter is a LiteOn 5208A with a 40° viewing angle [10]. This LED has a wavelength of 940 nm, a forward voltage of 1.2 VDC, and a Forward Current of 100 mA [10]. The LED Emitter is connected to a AAA battery using tape. The negative end of the IR LED is connected to the negative end of a AAA

**Fig. 2: IR Transmitter**



battery and the positive end is connected to the positive end.

Once connected the IR LED starts to transmit its signal, which is undetectable by human eyes, but is detectable by an IR Camera or a normal digital camera. This configuration is shown

in Figure 2.

### *C. Brian Peek's .NET Managed Library for Nintendo's Wiimote [5]*

Brian Peek's .NET Managed Library for Nintendo's Wiimote is a .NET Framework Library that was created to help .NET programs interface easily with a Wii Remote. This library can be used with either C# or VB.NET to create a program or application to access the Wii Remote's Features [5]. By allowing the world to interface with a Wii Remote I can use this managed library to help program my algorithm that will be used to help the Lukii locate itself. The library's API can be found at the author's Web Site and is a helpful tool in programming with the library [11].

### *D. Lukii*

The Lukii is the robot that I will be using in my experiment. It is a combination of a True Heroes© Radio Control Launching Tank and a Wii Remote. A Wii Remote will be attached to



the next emitter. The emitters were placed 4” off the ground, slightly lower than the Lukii’s Wii Remote, which is situated between 5” and 6” off the ground. This allows for the Lukii to detect when it is close to an emitter by looking at the y-axis angle between the Lukii and the emitter. When the angle dips too low the Lukii knows that it is near the emitter and can begin tracking the new emitter. The course setup allows for the Lukii to start underneath a desk, navigate around the desk and out of the room, navigate to the living room, and navigate around a couch to the final destination.

### C. The Algorithm

The following is the algorithm that was designed to allow the Lukii to navigate through our experiment and get to the end:

Sync Bluetooth Signal with Computer and Lukii

Loop

    Rotate Lukii's turret to acquire new distinct LED Signal

    Rotate Lukii's Base to line up with its turret

    While moving forward Loop

        Move Lukii forward until it loses signal or goes out of predefined sensor limits

        If out of sensor limit

            Rotate Lukii to reacquire signal

        End if

        Break when signal is out of y-axis predefined sensor limit

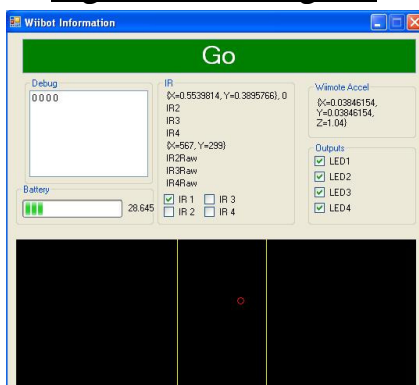
    End Loop

End Loop

//At this point the Lukii should be at the end of our experiment

### D. The Lukii Program

**Fig. 4: Lukii Program**



Using Brian Peek’s managed Wiimote Library I was able to create a program to display the information needed to navigate the Lukii through my experiment. The program is written in VB.NET and uses Windows Forms to display data to the Lukii operator. This program performs an array of

operations and displays them to the user. The source code is a combination of Brain Peek's WiimoteTest program, which helps facilitate the connection to the Wii Remote and to the Wiimote Library, as well as my Windows Forms which output the data that is needed to successfully navigate through the experiment. The program calculates the x-axis angle between the Lukii and an infrared transmitter, normalized to values between 0 and 1, to let the operator know where the Lukii is in relation to the emitter. It also informs the user if the Lukii needs to turn left or right, reacquire the emitter's signal, move on to the next emitter, or to just move forward towards the emitter. The user interface of the application is shown in Figure 4.

### *E. Lukii Program States*

As stated in the previous section the Lukii Program has different states associated with it to help it to achieve its goal. Described below are the five states of the Lukii Program. The five states are graphically represented in Figure 5.

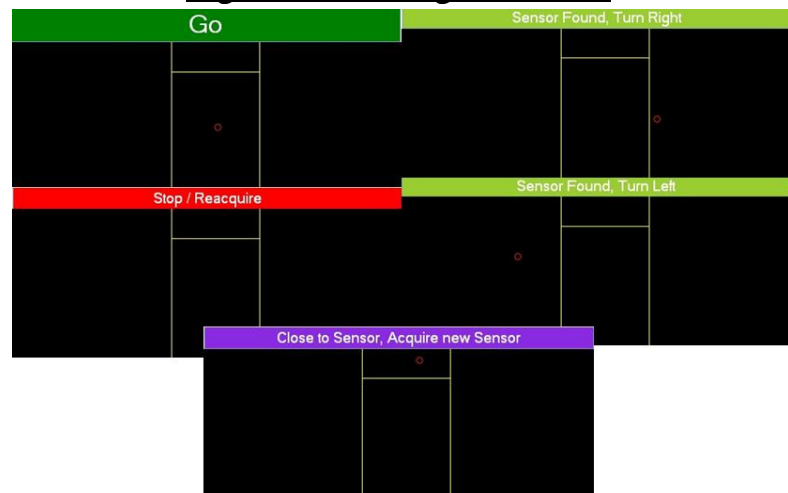
#### *E.1 Go State*

The go state of the Lukii is the state in which the IR transmitter has been acquired and the Lukii knows that it has a clear path to the transmitter. It continues to move forward until another state happens.

#### *E.2 Stop / Reacquire State*

The stop / reacquire state is the state of the Lukii when it is not receiving any IR transmissions. In this state the Lukii must reacquire a signal by moving its turret to the left or to the right until it has received an IR signal and the state has changed.

**Fig. 5: Lukii Program States**



### *E.3 Turn Left and Turn Right States*

These states happen when the Lukii acquires a signal but it is not lined up with the signal. These states let the Lukii know if it needs to turn left or right to line itself up with the sensor. Once the Lukii has aligned itself with the sensor the state changes back to the Go state.

### *E.4 Acquire New Sensor State*

When the Lukii is in the acquire new sensor state it has come close enough to the current sensor and needs to find another signal. The details behind this state have been explained previously.

## IV. RESULTS

Using the Lukii program I was able to successfully navigate the Lukii out from under my desk, through the door to the office, down the hallway to the living room, into the living room, around the couch, and to the finish line. While I was able to eventually achieve this there were many items that needed to be tweaked/changed to successfully navigate the Lukii through the course. The first problem that I ran into was the fact the Lukii is not able to make small adjustments to its orientation. While it is able to rotate 360 degrees without moving forward or backwards it does this rotation very quickly. This creates a problem as acquiring and reacquiring the emitters' signals sometimes requires very slight orientation changes. The turret is able to make slight orientation changes, but when trying to replicate the same movement with the Lukii's treads it over rotates every time. This did not take away from the final result of the experiment, but it did make it take longer to acquire every signal. Another item that needed to be changed was the range of values on the x-axis that constituted moving forward versus turning. I started out with a small range of values,  $.45 < x < .55$ . While these values did work it made it very difficult for the Lukii to be in the right orientation for moving forward. The small range of values made the Lukii

turn left, then right, then left and so on with very little time spent on moving forward. When the range of values was expanded to  $.40 < x < .60$  the Lukii spent much more time moving forward and a lot less time moving left and right. Overall the Lukii was successful in navigating through the course and locating itself in relation to the infrared transmitters.

## V. IMPROVEMENTS

By analyzing the test results I was able to come up with ideas to further improve the Lukii. While time restrictions made it impossible to implement these improvements they are given with the intent of good ideas for future work.

### *A. Distance from IR Transmitters*

The first improvement that I came up with was a way to tell how far the Wii Remote is from an IR sensor. While my implementation of using angles to let the Lukii know it is close to a transmitter does work it would be much easier if it could tell by just looking at the transmitter and determining how far away it is. The Wii Remote does have an IR report that tells the size of an IR transmitter [6]. Unfortunately this report is not implemented correctly in Brian Peek's .NET library and therefore I was unable to use it in my experiment. By fixing this feature and allowing the Lukii to tell the size of an IR transmitter it could be able to translate that size to a distance. Once translated this distance could be used to inform the Lukii how far it is from a certain transmitter and also help it in creating a sensor map.

### *B. Turret and Tread Alignment*

Another improvement idea that I came up with was finding a way for the turret to realign itself with the front of the Lukii. As stated before I had problems acquiring signals because the sensitivity of the treads was not sensitive enough for my experiment. The turret was sensitive enough but there was no automatic way for the turret to realign itself with the treads so that the

Lukii could move forward on the line that the turret was pointing. The improvement would be to find a way for the turret, when it finds the signal, to automatically align the Lukii with where the turret is pointing.

### *C. Differentiation of IR Transmitters*

Another idea to improve the Lukii is to find a way to differentiate one IR Transmitter from another. My experiment was set up in such a way that the Lukii could not reacquire the same signal that it just came from as the signal would always be right behind the Lukii. If it could detect more than one IR transmitter it would have a hard time figuring out which transmitter it had just come from and which transmitter it needed to go to. I have come up with two different ideas that address this problem. The first idea is to integrate a history into the Lukii, possibly using the accelerometer, to let the Lukii know which position each transmitter is in and if it is in the direction of one that it had previously visited. Another idea is to have each transmitter transmit different IR pulses so that the Lukii would know which transmitter was which by looking at the pulses.

### *D. Integration with Robot Localization Mapping Algorithms*

The last improvement, and the most important, is to fully integrate the idea of using the Wii Remote and the IR transmitters into a Robot Localization Algorithm. The way that this would be done would be to use the IR transmitters to make a sensor map that would help the Lukii identify where exactly it is. This sensor map would be created during a training phase and then tested to make sure that it is correct. Once correct the Lukii would be able to easily look at different IR signals and tell where it is. Technologies that may help with this integration of Robot Localization are discussed in the next section.

## VI. FUTURE TECHNOLOGIES

Because of the limited time and financial resources that were available there were many different technologies that I was thinking about implementing but was unable to do so. One main technology that I was unable to experiment with was the accelerometer in the Wii Remote and how it can be used for dead reckoning. The second technology that I was interested in experimenting with was adapting A.M. Ladd's et al. idea of using wireless Ethernet for robot localization [3] and adapting it to run on Bluetooth. Both ideas are further explained in the following sections.

### *A. Accelerometer and Dead Reckoning*

Keeping track of how much one moves by observing internal parameters without reference to the external world is known as dead reckoning [3]. While this idea of dead reckoning is normally implemented as an odometer in robot localization it is possible to adapt the idea of dead reckoning to harness the power of the accelerometer in the Wii Remote. We can produce a net displacement of the sum of motions from the accelerometer to keep track of how far the Lukii has traveled [3]. By using both dead reckoning and infrared sensors the Lukii will be better equipped to locate itself in its environment.

### *B. Bluetooth and Robot Localization*

As presented earlier A.M. Ladd et al. had an idea to use wireless Ethernet packets to have a computer locate itself in its environment. Since they proved their idea does work it makes sense to try to adapt their technology to the Wii Remote through Bluetooth. By adapting it to Bluetooth we would be able to use the Bluetooth capability of the Wii Remote to help locate the Lukii in its environment. By combining both the IR technology, dead reckoning, and wireless localization the Lukii would be a robot that would be very robust and could navigate easily in its environment.

## VII. CONCLUSION

In this paper we saw that a Wii Remote, when integrated with a robot, allows a robot to locate itself in relation to its surroundings and navigate through those surroundings to a predefined goal. We also saw that there are many different technologies out there that help with robot localization and the Wii Remote is able to be used with some of those technologies. We also saw that there are many features of the Wii Remote that can be adapted to help with robot localization. With time and resources a person could be able to fully integrate Wii Remote technology into a robot. My Lukii was able to use the information provided by a Wii Remote, in conjunction with a program using Brian Peek's .NET Wiimote Library, to successfully navigate the Lukii through multiple rooms and around multiple obstacles.

## REFERENCES

- [1] M. W. de Graaf, R. G. K. M. Aarts, J. Meijer, J. B. Jonker, "Ethernet-based communication framework for sensor integration on industrial robots," *International Conference on Robot Communication and Coordination (ROBOCOMM)*, no. 17, Oct. 2007.
- [2] T. Salter, F. Michaud, D. Létourneau, D. C. Lee, I. P. Werry, "Using Proprioceptive Sensors for Categorizing Human-Robot Interactions," *ACM SIGCHI/SIGART Human-Robot Interaction*, pp. 105-112, Mar. 2007.
- [3] A. M. Ladd, K. E. Berkis, A. Rudys, L. E. Kavraki, D. S. Wallach, "Robotics-based location sensing using wireless Ethernet," *Wireless Networks*, vol. 11, no. 1-2, pp. 189-204, Jan. 2005.
- [4] "Wii Remote - Wikipedia," October 2009. [Online]. Available: [http://en.wikipedia.org/wiki/Wii\\_Remote](http://en.wikipedia.org/wiki/Wii_Remote) [Accessed: Oct. 20, 2008].
- [5] "Coding4Fun : Managed Library for Nintendo's Wiimote," March 2007. [Online]. Available: <http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx> [Accessed: Nov. 5, 2008].
- [6] "Wiimote - WiiBrew," October 2008. [Online]. Available: <http://wiibrew.org/wiki/Wiimote> [Accessed: Oct. 21, 2008].
- [7] "WiiLi.org Wii Linux - Wiimote/Drivers," October 2008. [Online]. Available: <http://www.wiili.org/index.php/Wiimote/Drivers> [Accessed: Oct. 21, 2008].
- [8] Analog Devices, "Small, Low Power, 3-Axis +/-3 g iMEME Accelerometer," *Analog Devices*, 2007. [Online]. Available: [http://www.analog.com/static/imported-files/data\\_sheets/ADXL330.pdf](http://www.analog.com/static/imported-files/data_sheets/ADXL330.pdf). [Accessed: Nov. 20, 2008].
- [9] "Analog Devices: ADXL330," November 2008. [Online]. Available: <http://www.analog.com/en/mems-and-sensors/imems-accelerometers/adxl330/products/product.html> [Accessed: Nov. 20, 2008].
- [10] LITE-ON Electronics Inc., "LTE-5208A Data Sheet," *LITE-ON Electronics Inc.*, 2007. [Online]. Available: <http://sigma.octopart.com/145905/datasheet/Lite-On-LTE-5208A.pdf>. [Accessed: Nov. 20, 2008].
- [11] "WiimoteLib - .NET MAnaged Library for Nintendo Wii Remote - BrianPeek.com," November 2008. [Online]. Available: <http://www.brianpeek.com/blog/pages/wiimotelib.aspx> [Accessed: Nov. 6, 2008].