Michigan Tech
Create the Future

Michigan Technological University

Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's Reports - Open

Dissertations, Master's Theses and Master's Reports

2011

Tracking a non-cooperative target using a Doppler radar wireless sensor network

Michael K. Blaser
Michigan Technological University

Follow this and additional works at: https://digitalcommons.mtu.edu/etds

🎯 Part of the Electrical and Computer Engineering Commons

Recommended Citation

Follow this and additional works at: https://digitalcommons.mtu.edu/etds

🎯 Part of the Electrical and Computer Engineering Commons

Tracking a Non-Cooperative Target Using a Doppler Radar Wireless Sensor Network

By

Michael K. Blaser

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2011

This report, "Tracking a Non-Cooperative Target Using a Doppler Radar Wireless Sensor Network," is hereby approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE IN ELECTRICAL ENGINEERING.

Department of Electrical and Computer Engineering

Signatures:

Report Advisor
_____
Dr. Jindogn Tan

Report Co-Advisor
_____
Dr. Shiyan Hu

Department Chair
_____
Dr. Daniel Fuhrmann

Date _____

Contents

List of Figures

List of Tables

Abstract

Tracking or target localization is used in a wide range of important tasks from knowing when your flight will arrive to ensuring your mail is received on time. Tracking provides the location of resources enabling solutions to complex logistical problems. Wireless Sensor Networks (WSN) create new opportunities when applied to tracking, such as more flexible deployment and real-time information. When radar is used as the sensing element in a tracking WSN better results can be obtained; because radar has a comparatively larger range both in distance and angle to other sensors commonly used in WSNs. This allows for less nodes deployed covering larger areas, saving money.

In this report I implement a tracking WSN platform similar to what was developed by Lim, Wang, and Terzis [1]. This consists of several sensor nodes each with a radar, a sink node connected to a host PC, and a Matlab© program to fuse sensor data. I have re-implemented their experiment with my WSN platform for tracking a non-cooperative target to verify their results and also run simulations to compare. The results of these tests are discussed and some future improvements are proposed.

Chapter 1 Introduction

1.1 Motivation

Tracking or target localization is used in a wide range of important tasks from knowing when your flight will arrive to ensuring your mail is received on time. Tracking provides the location of resources enabling solutions to complex logistical problems.

Wireless Sensor Networks (WSN) allow for new opportunities when applied to tracking. More flexible deployment of a tracking network can be achieved since each sensor node is wireless. Together the WSN can provide real-time information on what it is tracking.

By using radar as the sensing element in the tracking WSN better results can be obtained. Radar has a comparatively larger range both in distance and angle to other sensors commonly used in WSNs. This allows for less nodes deployed covering larger areas, saving money. This introduces many new possible applications such as intruder detection or elderly persons monitoring.

Intruder detection when applied to WSNs is the tracking of an invader into the networks domain [2]. This has been a major topic of interest for the military to deploy on battle fields but could also be applied to simple building security. By implementing intruder detection with a WSN it greatly automates the task and allows for real-time information.

Another application of tracking in WSNs is to monitor the elderly [3]. The elderly, in partially assisted living, wish to keep their independence but want the security of knowing someone can help them. By using a WSN to track a patient it is possible to determine if they have fallen or are in need of assistance, and allows for up to the date information on their location.

1.2 Problem Specification

Tracking in WSNs can be broken into three sub problems which apply to all WSNs; sensing, networking, and data fusion. Quality data must be obtained from each sensor node to provide accurate results. Each node must be networked together so data and commands can be communicated to and from each node. All the data from the WSN should be somehow combined to provide better information to our tracking goal.

1.3 Contributions

In this report a tracking WSN platform similar to what was developed by Lim, Wang, and Terzis is implemented [1]. Our platform consists of several sensor nodes each with a radar, a sink node connected to a host PC, and a Matlab© program to fuse sensor data. We have re-implemented their experiment with our WSN platform for tracking a non-cooperative target to verify their results and simulations to compare. The results of these tests are discussed and some future improvements are proposed.

1.4 Organization

The remainder of this report is organized as follows:

- Chapter 1 – Introduction
- Chapter 2 – This section explores previous work done on tracking with WSNs for both cooperative and non-cooperative methods.
- Chapter 3 – This section gives a general approach of the techniques and methods I use to create a non-cooperative tracking WSN.
- Chapter 4 – This section describes in detail the hardware and software used to implement the WSN.
- Chapter 5 – This section describes the tests and simulations performed to evaluate the WSN and show their results.
- Chapter 6 – This section analyzes the results from the tests, discusses these results, and shows some of the limitations of the WSN.
- Chapter 7 – This section looks at several methods that could be used to reduce the limitations found from the experiments for future work.
- Chapter 8 – Conclusion

Chapter 2 Related Work

Tracking using a WSN is by no means a new concept and some of the first to explore it were Bahl and Padmanabhan in the year 2000 [4]. Their WSN used received signal strength (RSS) from a sensor on the target and multiple distributed sensors to track people through a building. Since the target is an active element in the tracking algorithm this method is considered cooperative tracking. The majority of research on tracking with WSNs has been done with cooperative tracking.

2.1 Cooperative Tracking

The breadth of cooperative tracking research is quite large so the following are just so distinct examples of the area. In "Simultaneous Localization and Mobile Robot Navigation in a Hybrid Sensor Network" the authors explore using a mobile robot to simultaneously localize both the robot and the WSN without an explicit distance sensor [5]. They use radio messages which have some range in combination with the moving robots odometry and by moving in and out of the range of the sensors allows for both the robot and the sensors to localize. The well known Cricket WSN platform developed by Priyantha, Chakraborty, and Balakrishnan used both radio and sound wave in a time difference of arrival method to localize nodes [6]. Lastly "Tracking Mobile Nodes Using RF Doppler Shifts" measures the Doppler shift of a node's radio beacon to other stationary nodes [7]. All of these examples are limited to have a node on the tracking target and this is impractical for some applications such as intruder detection.

2.2 Non-Cooperative Tracking

Non-cooperative tracking attempts to solve the tracking problem without the target actively participating. This makes tracking problem more difficult since it puts more importance on the sensing element. Therefore the different

approaches to non-cooperative tracking are based around different sensing technologies.

Ultrasonic sensors provide accurate, semi-directional information. Nishida, et al. use ultrasonic sensors embedded into the ceiling of a roof to localize humans based on head positions [8]. This allows for the minimally invasive tracking people down to 5cm in a building but is limited to only people. Li, et al. attempts to track generic moving objects with sensor "poles" which have multiple ultrasonic sensors at different angles on them [9]. This allows for more flexible tracking but requires poles in a grid spaced 2.5m apart.

Camera based WSNs allow for the greatest flexibility with large amounts of possible tracking information but are the processing intensive. Sanchez-Matamoros, Dio, and Ollero used vision-based WSN to detect, localize, and track objects [10]. Their solution focused on processing the images and reducing the needed network bandwidth. The energy need to process images is prohibitive to the lifetime of the WSN.

While ultrasonic and camera based WSN are more common there are still many other techniques. He, et al. implemented a large scale WSN with 200 nodes to track vehicles over a 300m and 200m stretch of road called VigilNet [11]. The sensing mechanism combined multiple low fidelity, low cost sensors; acoustic, photo, temperature, and passive infrared. This allows for a binary detection scheme which provides low resolution but great scalability as 200 nodes covered a 5000m² area. Mao, et al. uses a creative solution involving room light fixture and with groups of extremely low cost light sensors to track a person moving down a hall [12]. While the sensors were very low cost they were still able to localize a target to within 20cm but needed 40 sensors to cover one 12.5m hallway.

Chapter 3 Approach

We divide the non-cooperative tracking problem up into two part, sensing and tracking. The distributed WSN nodes collect information and then send it to a sink node connected to a host PC as seen in Figure 3.1. Software on the Host PC fuses all of the sensor information, tracks the target, and provides real-time output.
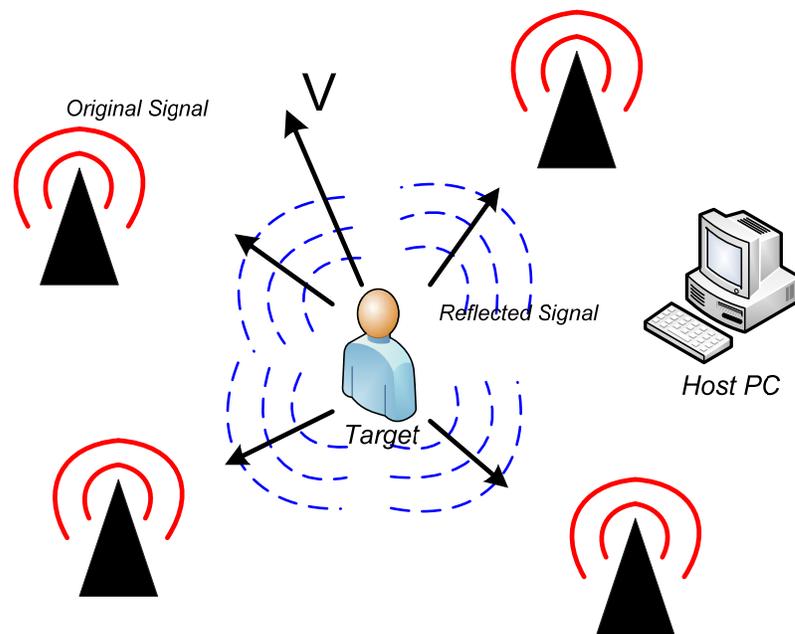
Figure 3.1: General layout of sensor node radars in reference to the target.

3.1 Sensing

Each WSN sensor node has a BumbleBee radar to collect Doppler frequency information from the target movement. This information is then processed with

autocorrelation to find the target's Doppler frequency using Equation (3.1) and Equation (3.2)

$$R_n(1) = \sum_n x_n x_{n-1}^*$$

(3.1)

$$\angle R_n(1) = \tan^{-1} \frac{im\{R_n(1)\}}{re\{R_n(1)\}}$$

(3.2)

Autocorrelation will give the average frequency from the Fourier Transform of the Doppler information, and because there is only one target within the WSN field this should be the Doppler shift frequency and by applying Equation (3.3) we can calculate the velocity of the target.

$$V_d = \frac{f_d \lambda}{2}$$

(3.3)

After the velocity is calculated it is sent off the sink node over the WSN as a UDP packet. The sink node collects all UDP packets from the sensor nodes.

3.2 Tracking

The sink node receives all incoming UDP packets from the sensors taking each and transmits them to the host PC as a string message. The host PC runs Matlab© which reads in the messages, feeds them into an Extended Kalman Filter (EKF), and outputs the current estimation of the EKF.

3.2.1 Extended Kalman Filter

The EKF is used over a traditional Kalman Filter because the observation model is non-linear; although the state transition model is linear. Shown in Table 3.1 is the EKF algorithm used in our research.

6

Table 3.1: Extended Kalman Filter Algorithm.

Priori
$$\hat{x}_k^- = A\hat{x}_{k-1}$$
$$P_k^- = AP_{k-1}^- A^T + Q$$

Posteriori
$$K_k = \frac{P_k^- J_k^T}{J_k P_k^- J_k^T + R}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - F(\hat{x}_k^-))$$
$$P_k = (\mathbb{I} - K_k J_k)P_k^-$$

The state transition model assumes the target moves at a constant velocity and is shown below as matrix A, with the state vector X in Equation (3.4). Note p_x, p_y and v_x, v_y and the position and velocity of the target respectively.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \hat{x}_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \tag{3.4}$$

The observation model is based on the geometric position of the radar and the target state vector, $\hat{x}_k$, as shown in Figure 3.2

7
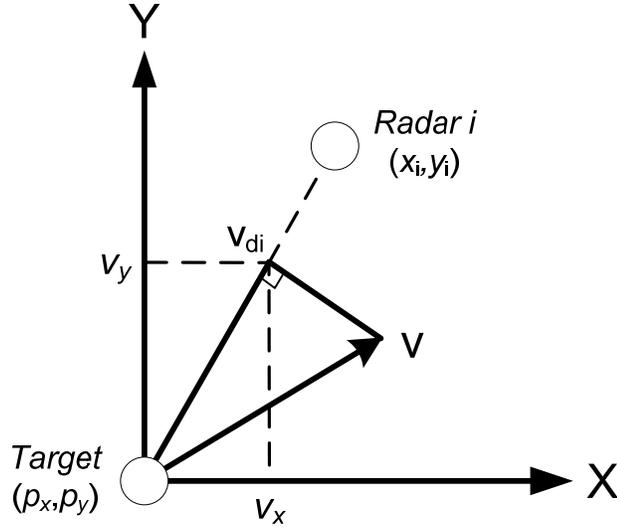
Figure 3.2: Shows the radar and target layout.

The observation model, which takes a state vector and gives the expected measurements, is described in Equation (3.5). Each F_i represents an expected sensor node measurement, the Doppler velocity, shown as v_{di} in Figure 3.2

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \end{bmatrix}, F_i = \frac{v_x(x_i - p_x) + v_y(y_i - p_y)}{\sqrt{(x_i - p_x)^2 + (y_i - p_y)^2}} \tag{3.5}$$

To calculate the Kalman gain matrix K_k the Jacobian matrix, J_k, of the observation model F is show in Equations (3.6)-(3.10)

$$\frac{\partial F_i}{\partial p_x} = \frac{v_x(y_i - p_y)(x_i - p_x) - v_y(x_i - p_x)^2}{((x_i - p_x)^2 + (y_i - p_y)^2)^{\frac{3}{2}}} \tag{3.6}$$

$$\frac{\partial F_i}{\partial p_y} = \frac{v_y(x_i - p_x)(y_i - p_y) - v_x(y_i - p_y)^2}{((x_i - p_x)^2 + (y_i - p_y)^2)^{\frac{3}{2}}} \tag{3.7}$$

$$\frac{\partial F_i}{\partial v_x} = \frac{(x_i - p_x)}{\sqrt{(x_i - p_x)^2 + (y_i - p_y)^2}} \tag{3.8}$$

$$\frac{\partial F_i}{\partial v_y} = \frac{(y_i - p_y)}{\sqrt{(x_i - p_x)^2 + (y_i - p_y)^2}} \tag{3.9}$$

$$J_k = \nabla F = \nabla \begin{bmatrix} F_1 \\ \vdots \\ F_i \end{bmatrix} = \begin{bmatrix} \dfrac{\partial F_1}{\partial p_x} & \dfrac{\partial F_1}{\partial p_y} & \dfrac{\partial F_1}{\partial v_x} & \dfrac{\partial F_1}{\partial v_y} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial F_i}{\partial p_x} & \dfrac{\partial F_i}{\partial p_y} & \dfrac{\partial F_i}{\partial v_x} & \dfrac{\partial F_i}{\partial v_y} \end{bmatrix} \tag{3.10}$$

The noise covariance matrices Q and R for the state transition model and observation model respectively are shown below in Equation (3.11)

$$Q = \begin{bmatrix} 50 & 0 & \Delta t & 0 \\ 0 & 50 & 0 & \Delta t \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}, R = \begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix} \tag{3.11}$$

Chapter 4 Implementation

Show below in Figure 4.1 is the system block diagram of my implementation to the WSN tracking platform from Chapter 3.
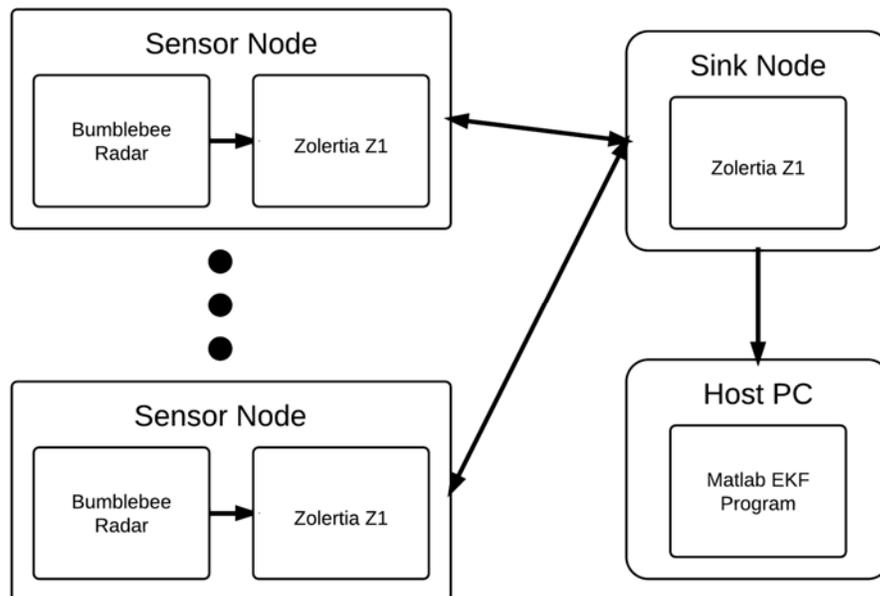
Figure 4.1: System Block Diagram of the tracking WSN.

4.1 Sensing

4.1.1 Hardware

The sensing hardware is implemented on two WSN node platforms the Tmote Sky mote and Zolertia Z1 shown in Figure 4.2 [13, 14]. Each sensor node and sink node can use either platform due to the portability of the Contiki software described in the next section. As mentioned in Chapter 3 the BumbleBee Radar

by The Samraksh Company is connected to each sensor node [15]. Shown in Table 4.1 is the wiring for the Zolertia Z1 platform to the BumbleBee radar.
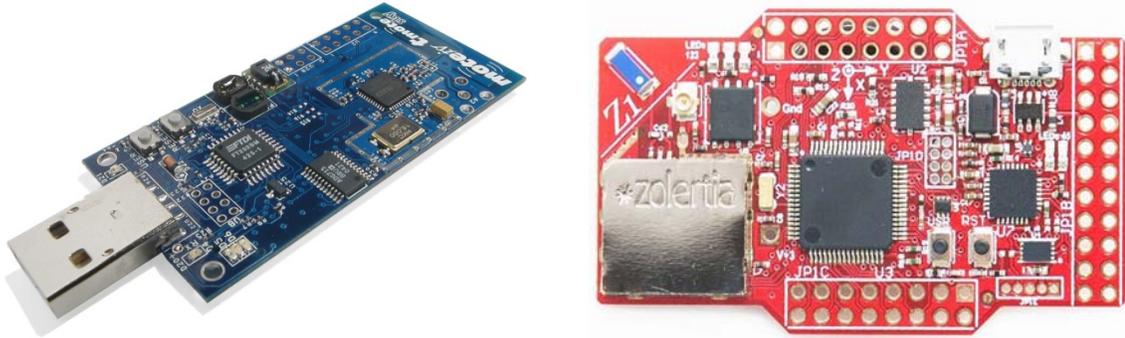
(a) Tmote Sky   (b) Zolertia z1

Figure 4.2: The two WSN hardware Platforms used.

Table 4.1: Wiring for Sensor Node.

BumbleBee Radar	Zolertia Z1	Battery Pack
J1-1		Ground
J1-2		Power
J1-3		Power
J1-4	JP1A-2	
J2-1	JP1A-1	
J2-2	JP1A-7	
J2-4	JP1A-9	

4.1.2 Software

Each sensing node and the sink node runs Contiki an embedded OS. Contiki was chosen for its small footprint and large feature set. Contiki uses an event driven multi-tasking scheduler using a unique programming construct the Protothread. Protothreads are a memory efficient, stackless version of a thread that use cooperative scheduling allowing straight forward coding even with multiple

threads. Contiki also provides a lightweight IP stack library to ease implementation of networking.

The client.c program shown in Appendix A runs on each sensor node and behaves as shown below in Figure 4.3. The first event thread simple initializes the sensor node by configuring the radar driver, creating the UDP network connections, and starting the periodic timer. The second event thread is based around the periodic timer which creates a paced loop that reads the radar sensor and computes the autocorrelation. The last two event threads are used to receive and then respond to beacon messages from the sink node.
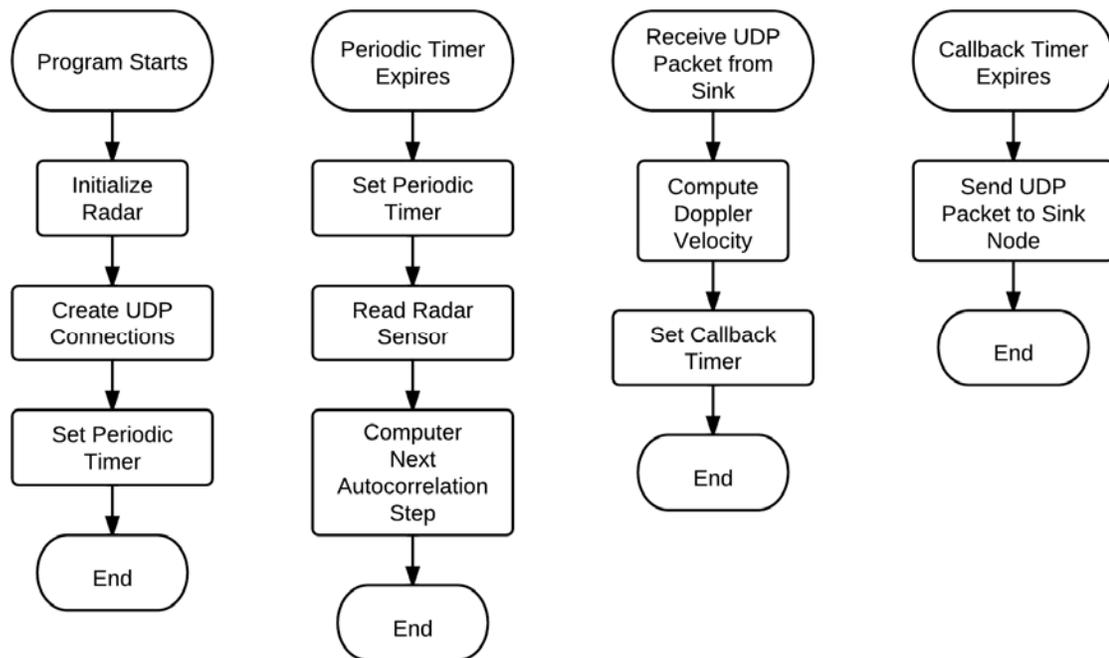
Figure 4.3: Block Diagram of client.c code flow.

The sink node runs the host.c program shown in Appendix A and is described below in Figure 4.4. In the sink node the first event thread initializes by creating the UDP network connections and starting the periodic callback timer for the beacon. The second event thread sends the periodic beacon messages over the UDP connection to all sensor nodes. The last event thread receives data from the sensor nodes and prints it to the serial link sent to the host PC.
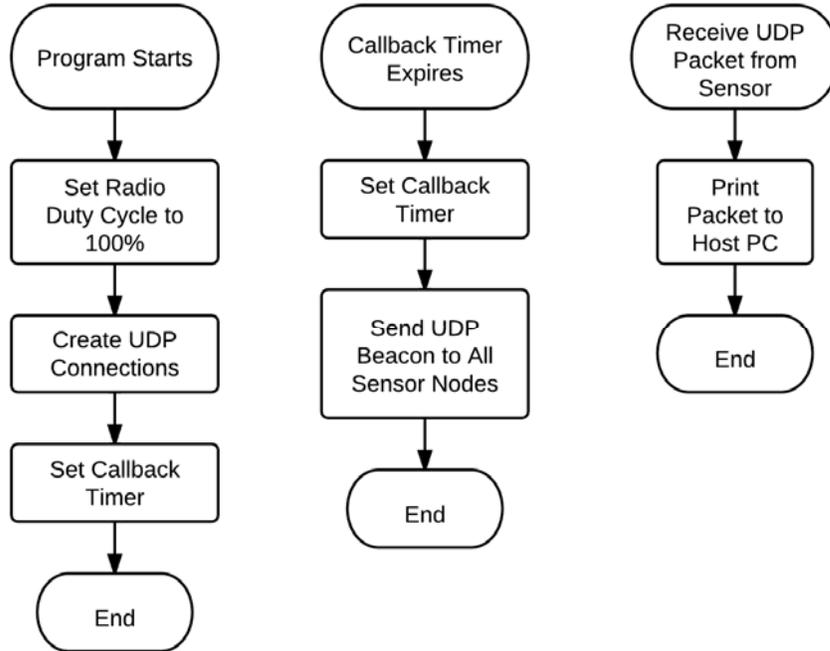
Figure 4.4: Block Diagram of host.c code flow.

4.2 Tracking

The host PC that is running Matlab© and performs the tracking is a Pentium4 2.6GHz with 1G ram. The Matlab© code tracker.m is shown in Appendix A and behaves as shown in Figure 4.5. The program starts by initializing the EKF parameters and setting up the serial link to the sink node. The serial link is then read and sensor data is matched into set corresponding to each beacon. Since the beacon reaches each sensor at approximately the same time this beacon set has the time synchronous readings of each sensor radar. Once a set is completed it is run through the EKF and the results are displayed. Since the UDP network is best effort not all messages are guaranteed to be received. The time step for the EKF is dynamic and is iterated by the time difference between two complete beacon sets. This allows the tracking to be robust even in a lossy WSN.
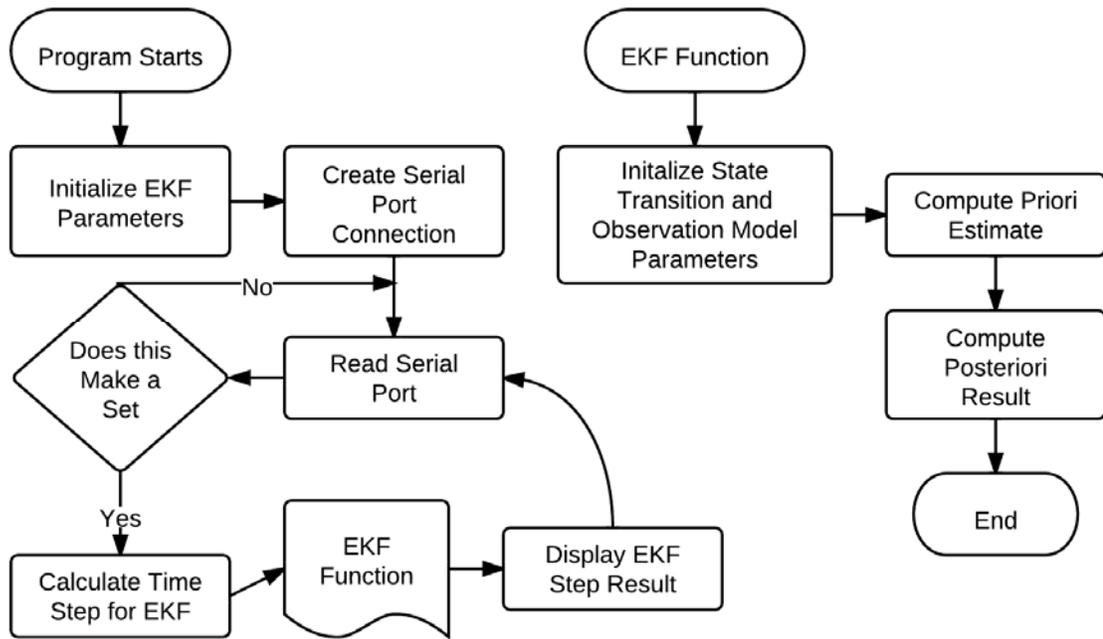
Figure 4.5: Block diagram of tracker.m program flow.

Chapter 5 Evaluation

To evaluate the accuracy of the implementation a simulation was and simple test bed were setup to evaluate performance of the system. Shown in Figure 5.1 is the configuration used for both the simulation and tests, which shows the two sensors 6' apart with the sink node in the middle all collinear.
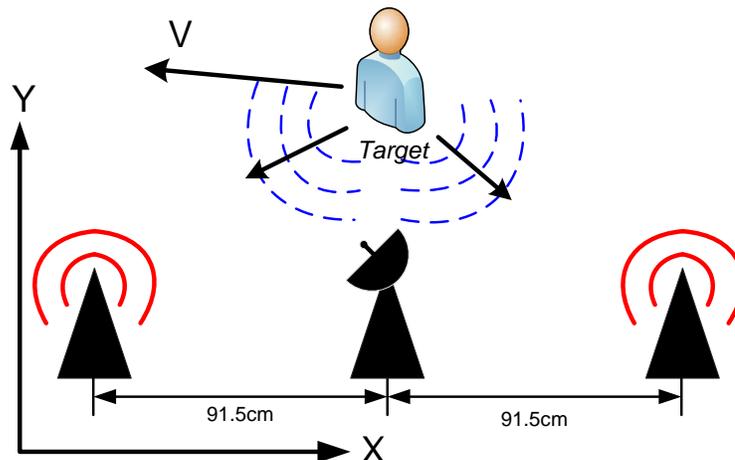
Figure 5.1: Diagram of experiment and simulation layout.

5.1 Experiment

Both radars' ranges were adjusted to full-scale (10m) and then a single subject moved in the WSN in a predetermined pattern so the ground truth was known. The sink node was set to send beacon messages every 0.5 seconds. The tests were performed in a furnished graduate student office with only the subject moving at approximately 30cm/s. Three different test patterns were used; a line perpendicular to the two sensors, an oblique line, and a diamond. Shown in

Figure 5.2 are the test results for the perpendicular pattern.
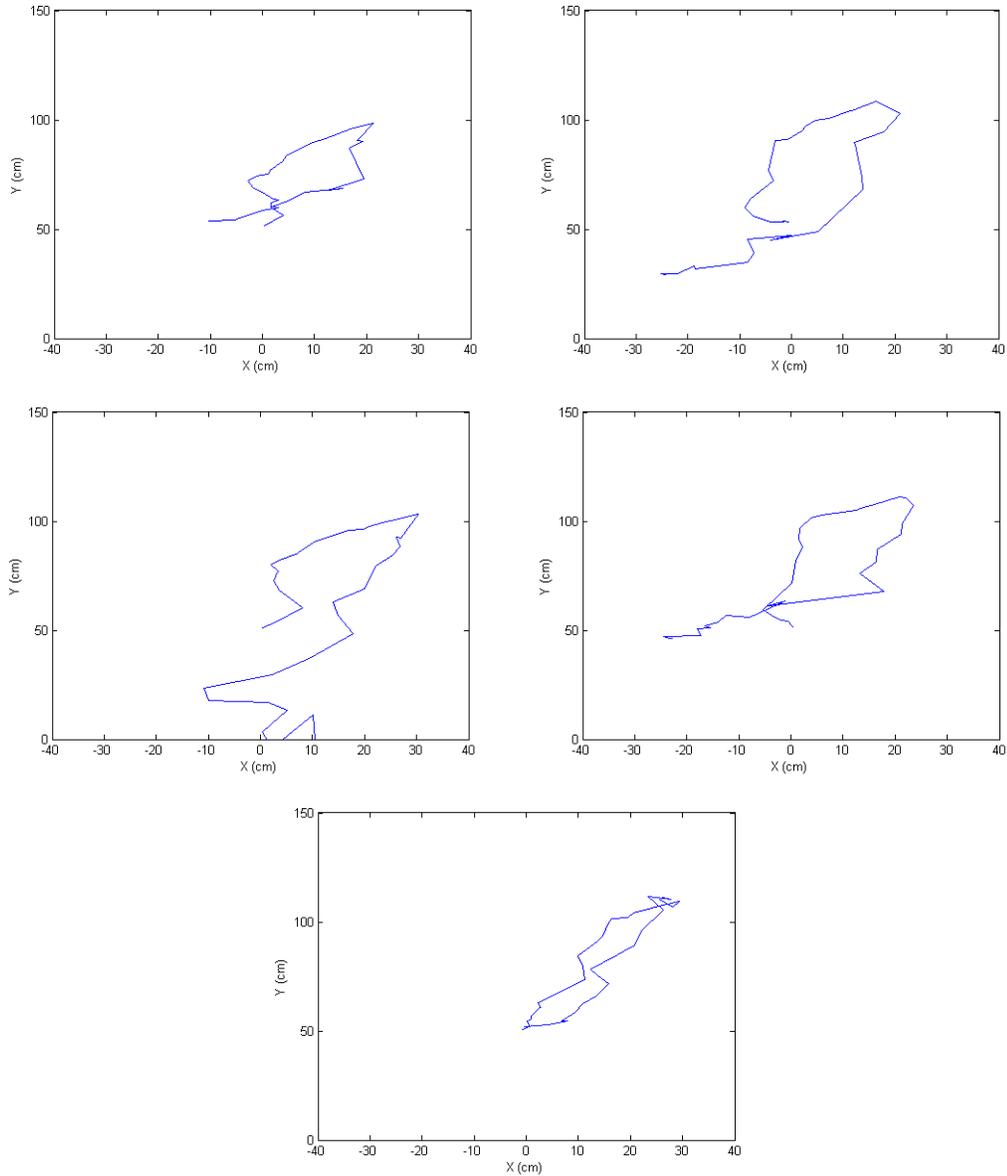
Figure 5.2: Experimental results for the perpendicular test pattern.

The perpendicular test pattern was created by starting 50cm back from the x axis and on the y axis. The subject then moved 100cm forward and then 100cm back to the starting position. It can be seen from

Figure 5.2 the EKF generally tracked the trajectory of the subject was close to linear in shape. In all five trials the EKF does not track the subject to the expected 150cm but does generally finish near the place it started. Shown in Figure 5.3 are the results from the oblique line test trials.

16
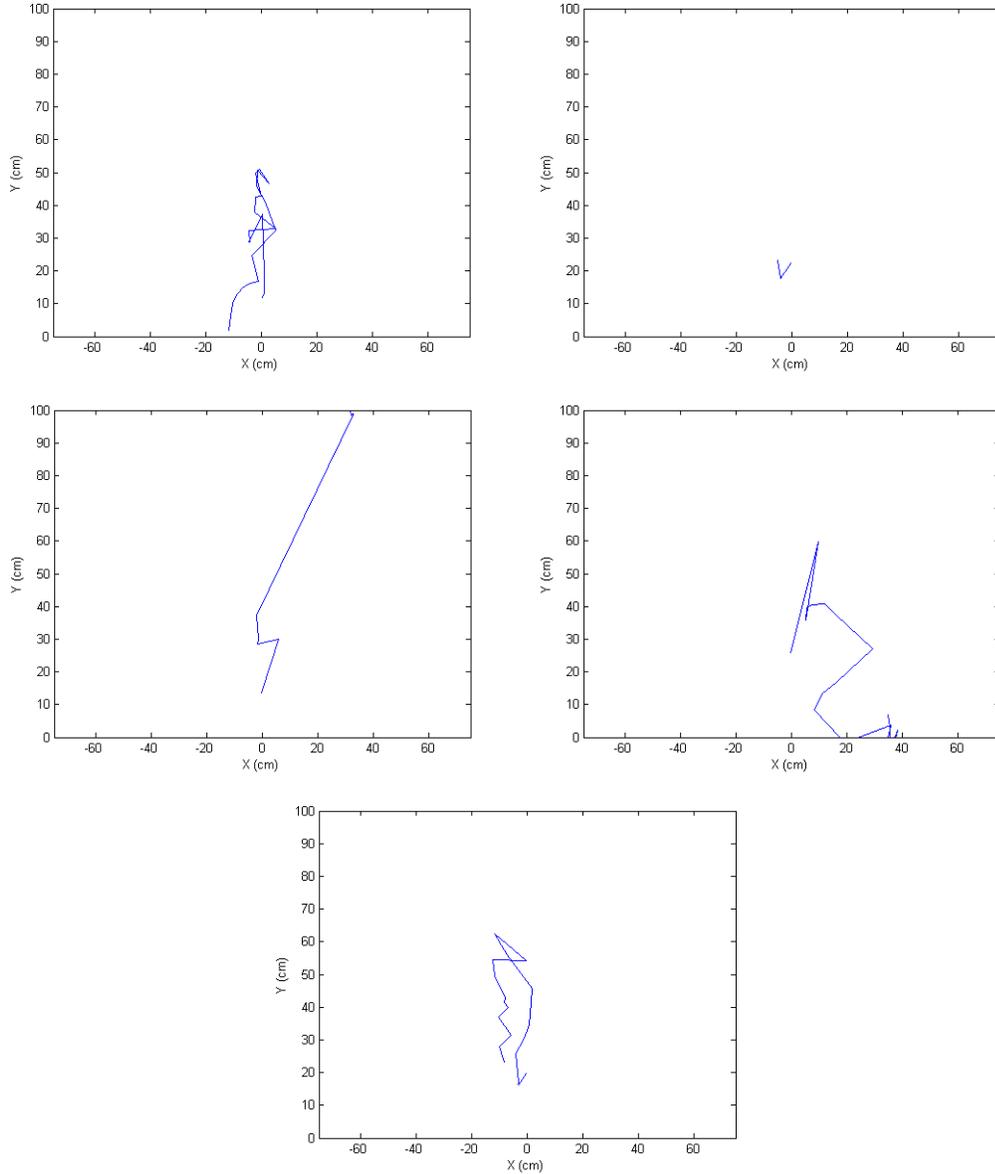
Figure 5.3: Experimental results for the oblique test pattern.

The oblique test pattern is made by the subject starting 10cm from the x axis and on the y axis. The subject then moves 75cm along the y axis and -75 along the x axis to create an oblique line to the x and y axes. All trials have a general linear shape but again do not track all the way to the far point of the test. I attribute this to the noise threshold filter in the EKF which is set to 5cm/s. Shown in Figure 5.4 are some of the trials from the diamond test pattern.

17
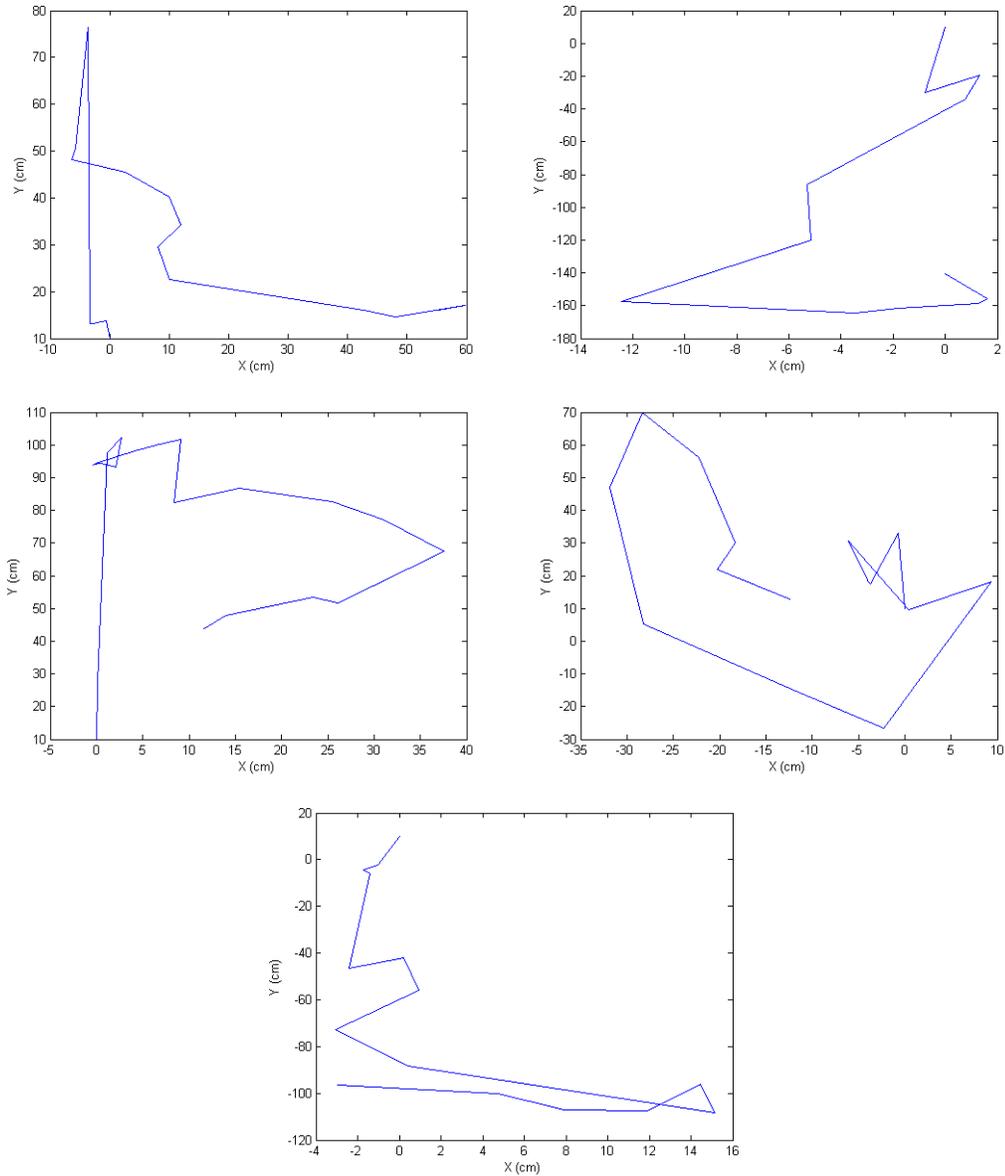
Figure 5.4: Experimental results for the diamond test pattern.

To generate the diamond test pattern the subject started 10cm from the x axis and on the y axis, same as the oblique pattern tests. The subject then moved in 140cm by 140cm segments to create a square diamond pattern. The failure of the EKF is apparent from the lack of a diamond shaped pattern and will be discussed in Chapter 6.

5.2 Simulation

Shown in Figure 5.5 are the simulation results for the perpendicular test pattern.
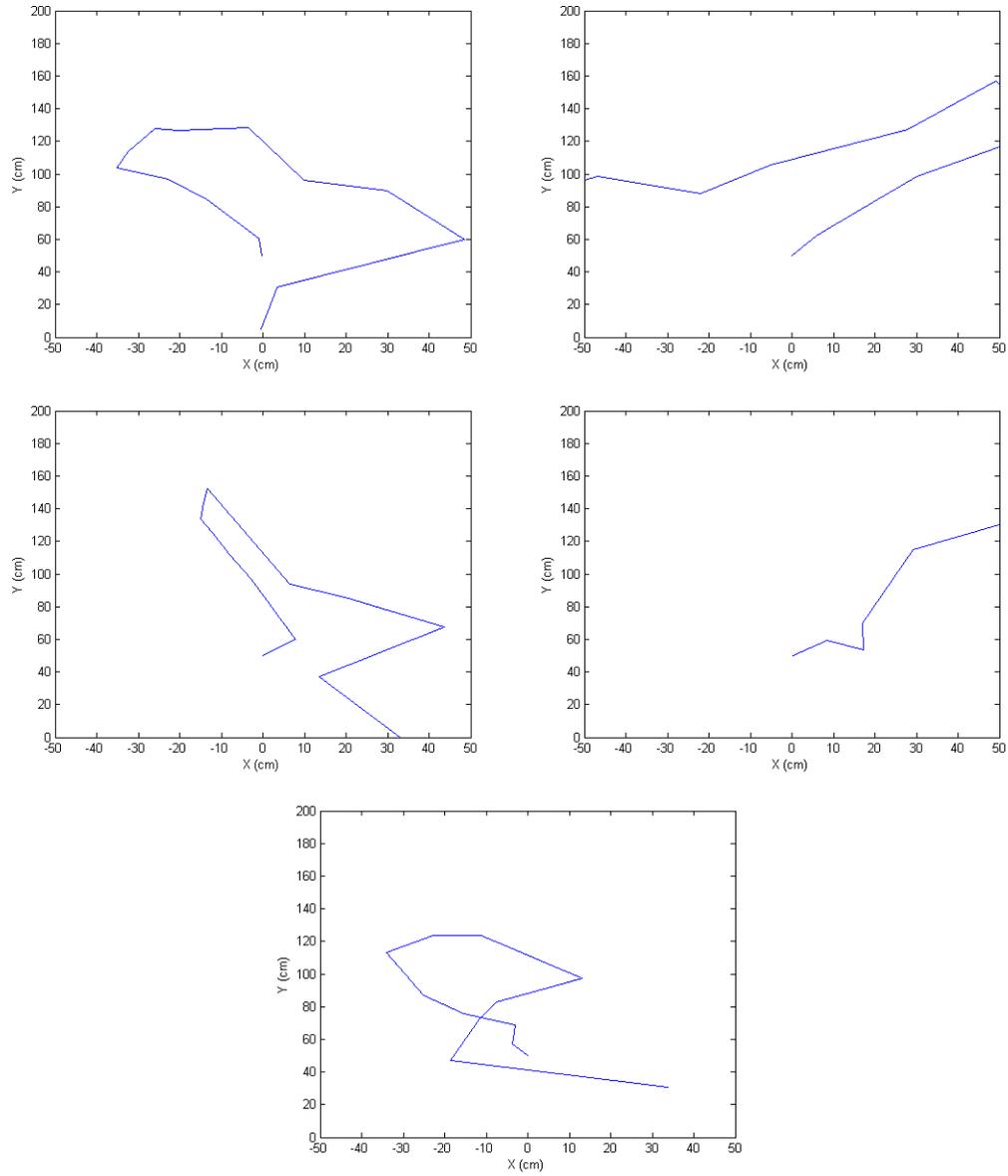
Figure 5.5: Simulation results of perpendicular test pattern.

These results are visually comparable to the experiment results for the perpendicular test pattern. The results for the oblique test pattern also show about the same performance as seen from the experimental test results.
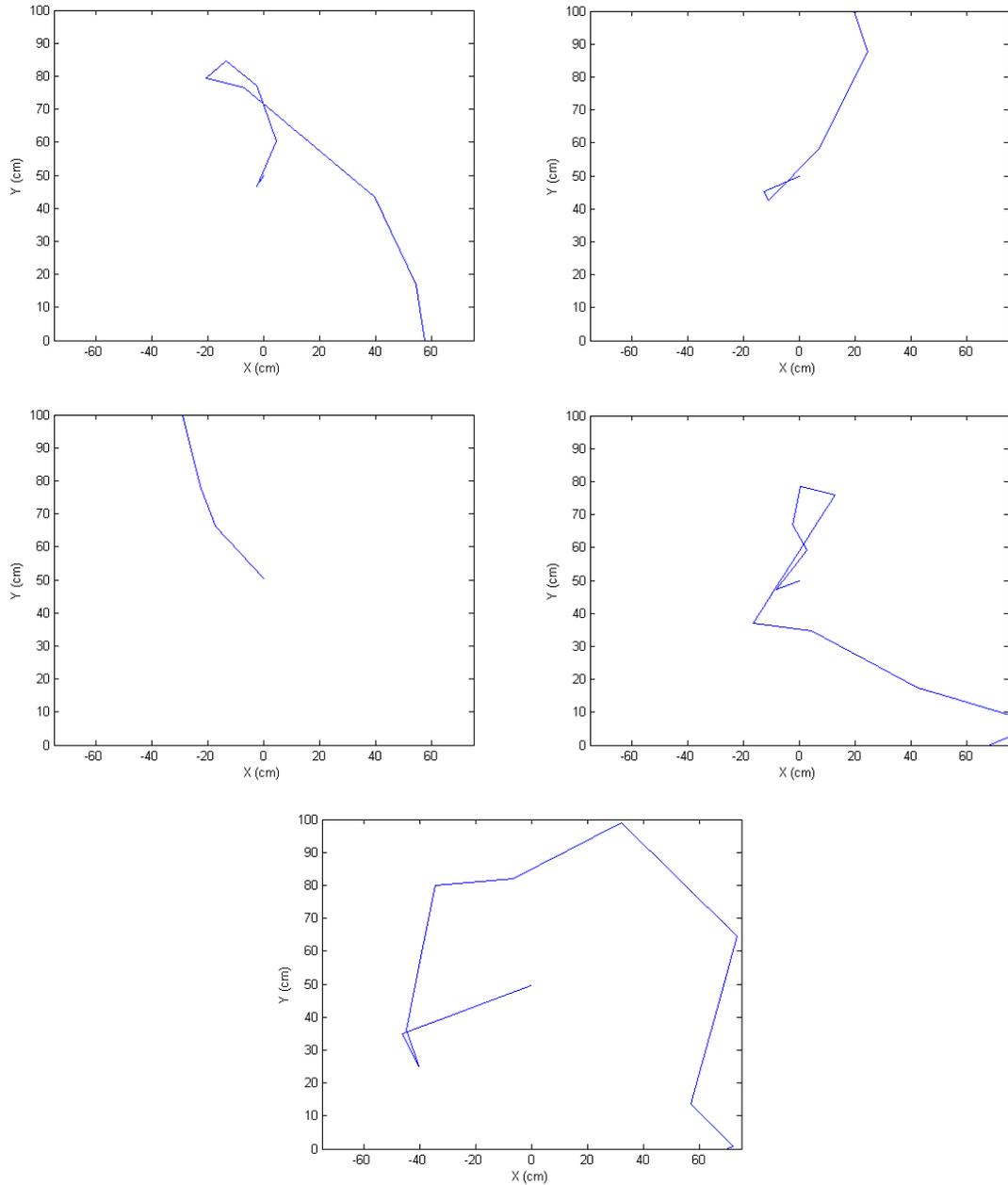
Figure 5.6: Simulation results of oblique test pattern.

Chapter 6 Discussion

Although the EKF runs in real-time each step was logged for data analysis. Shown in Table 6.1 are the statistical results from the experiment for the perpendicular test pattern.

Table 6.1: Statistical summary for experimental perpendicular test pattern.

Trial	1	2	3	4	5	Average	Std Dev
Average Error (cm)	30.3	33.6	39.8	28.5	24.9	31.4	5.6
Max Error (cm)	56.9	59.9	66.4	57.3	49.6	58.0	6.1

Following the results from Table 6.1 we are 95% confident the average error of the tracking algorithm is within 31.4±11.2cm, which gives a resolution of approximately 0.5 meters. Furthermore, we are 95% confident the maximum error is 58.0±12.2cm. The maximum error is also approximately 0.5 meters.

Table 6.2: Statistical summary for experimental oblique test pattern.

Trial	1	2	3	4	5	Average	Std Dev
Average Error (cm)	42.8	43.7	94.2	72.2	39.0	58.4	24.0
Max Error (cm)	89.9	97.7	126.0	116.1	86.3	103.2	17.2

The results from the oblique test pattern are close to the perpendicular but with higher averages and larger standard deviations. We are 95% confident the average error is 58.4±48.0cm. This resolution is approximately 1 meter. We are

95% confident the maximum error is 103.2±34.4cm, which is close to the 1 meter but we are much less certain.

The diamond test was not worth statistically analyzing because it can be seen from Figure 5.4 the algorithm did not track a diamond shape. This is because of the constant velocity assumption and low sample rate of the sensor node by the beacon. The largest limitation of this WSN setup is the time step of the EKF. The current network setup cannot reduce the time step of the beacon to less than half a second. Any faster and the network saturates with messages causing messages to be lost and/or received out of order. This would cause unreasonable delays to the real-time aspect of the EKF.

Chapter 7 Future Work

Even though the time step of the current implementation places limits on the performance of tracking there are other ways to gain performance such as changing the tracking algorithm.

The plain EKF was implemented out of simplicity and several other versions exist such as the unscented EKF. Even better than an EKF would be to implement a Particle Filter which would better represent the non-linear observation model leading to higher resolutions.

The current setup can only track a single target within its domain and this leads to a simpler implementation but limits the applications of the platform. Instead of using autocorrelation to determine the Doppler velocity of a target a Fast-Fourier Transform (FFT) could be used. Although this is much more computationally intensive it would allow for the tracking of many targets at one time. Our current EFK cannot work in conjunction with the FFT method because it only supports a single position hypothesis but a multi-hypothesis tracking algorithm such as a Particle Filter could be used. Multi-target tracking is a desirable feature and possible with the current hardware.

Chapter 8 Conclusion

The goal of this report was to implement a tracking WSN platform similar to what was developed by Lim, Wang, and Terzis (1). A WSN was developed using the Tmote Sky, Zolertia Z1, and a host PC. Each sensor node consisted of a BumbleBee radar connected to a Zolertia Z1, which sent its data to a sink node. The sink node was a Tmote Sky connected to a host PC through a serial link. The host PC ran a Matlab© program that read in the sensor data from the serial link and fused it through an EKF.

An experiment was setup to verify the WSN platform. Several test target movement patterns were used and the results logged for further analysis. It was found that simple linear motion could be tracked but any complex motion with numerous changes in velocity or direction failed to be tracked. This can be attributed to the state transition model assumption of constant velocity and the large time step size of the EKF. Results could be improved by changing the tracking algorithm to a different EKF such as an unscented EKF or using a Particle Filter to better model the observation model. Furthermore, our research was limited to single target tracking but could be expanded to multi-target tracking if autocorrelation was replaced with a FFT and using a multiple hypothesis tracking algorithm such as a Particle Filter.

Works Cited

[1]   L. Jong Hyun, I. J. Wang, and A. Terzis, "Tracking a non-cooperative mobile target using low-power pulsed Doppler radars," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, 2010, pp. 913-920.

[2]   W. Yun, W. Xiaodong, X. Bin, W. Demin, and D. P. Agrawal, "Intrusion Detection in Homogeneous and Heterogeneous Wireless Sensor Networks," *Mobile Computing, IEEE Transactions on,* vol. 7, pp. 698-711, 2008.

[3]   T. Ali Maleki, K. Arezou, and A. Hamid, "Smart home care network using sensor fusion and distributed vision-based reasoning," in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks* Santa Barbara, California, USA: ACM, 2006.

[4]   P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2000, pp. 775-784 vol.2.

[5]   S. Shenoy and T. Jindong, "Simultaneous localization and mobile robot navigation in a hybrid sensor network," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 1636-1641.

[6]   B. P. Nissanka, C. Anit, and B. Hari, "The Cricket location-support system," in *Proceedings of the 6th annual international conference on Mobile computing and networking* Boston, Massachusetts, United States: ACM, 2000.

[7]   K. Branislav, L. Akos, and K. Xenofon, "Tracking mobile nodes using RF Doppler shifts," in *Proceedings of the 5th international conference on Embedded networked sensor systems* Sydney, Australia: ACM, 2007.

[8]   Y. Nishida, T. Hori, S. Murakami, and H. Mizoguchi, "Minimally privacy-violative system for locating human by ultrasonic radar embedded on ceiling," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, 2004, pp. 1549-1554 vol.2.

[9]   L. Hongbin, M. Di, C. Jiming, S. Youxian, and S. Xuemin, "Networked Ultrasonic Sensors for Target Tracking: An Experimental Study," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 2009, pp. 1-6.

[10]   J. M. Sanchez-Matamoros, J. R. M. d. Dios, and A. Ollero, "Cooperative localization and tracking with a camera-based WSN," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, 2009, pp. 1-6.

[11]   H. Tian, P. Vicaire, Y. Ting, L. Liqian, G. Lin, Z. Gang, R. Stoleru, C. Qing, J. A. Stankovic, and T. Abdelzaher, "Achieving Real-Time Target Tracking UsingWireless Sensor Networks," in *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*, 2006, pp. 37-48.

[12]   M. Xufei, T. ShaoJie, X. Xiaohua, L. Xiang-Yang, and M. Huadong, "iLight: Indoor device-free passive tracking using wireless sensor networks," in *INFOCOM, 2011 Proceedings IEEE*, 2011, pp. 281-285.

[13]   "Tmote Sky Ultra low power IEEE 802.51.4 compliant wireless sensor module," Moteiv Corperation, 2006. Available from http://www.sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf

[14]   "Z1 Datasheet," Zolertia, 2010. Available from http://zolertia.com/sites/default/files/Zolertia-Z1-Datasheet.pdf

[15]   "Users manual for the BumbleBee (Model 0)," The Samraksh Company, 2008. Available from http://www.samraksh.com/docs/BumbleBee-UM-v103.pdf

Appendix A Software

client.c

```c
/*
 * client.c
 * Michael K. Blaser
 * Fall 2011
 *
 * This program is meant to be run on all sensor nodes.
 */

/*
 * Contiki OS BSD License
 *
 * Copyright (c) 2011, Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the Institute nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#include "contiki.h"
#include "lib/random.h"
#include "sys/ctimer.h"
#include "net/uip.h"
#include "net/uip-debug.h"
```

```
#include "simple-udp.h"

#include "dev/radar-sensor.h"

#define ABS(x) (x < 0 ? -x : x)

#include <stdio.h>
#include <string.h>

//This port is used to send messages to the sink node
#define UDP_PORT 1234
//This port is used to receive beacon messages from the sink node
#define UDP_PORT_BROAD 4321

//The BACKOFF_TIME is randomized every time the timer is set
#define BACKOFF_TIME    (random_rand() % (CLOCK_SECOND/4))

static struct simple_udp_connection unicast_connection;
static struct simple_udp_connection broadcast_connection;

static struct ctimer backoff_timer;

//Temporary variables used for the delayed send
static const uip_ipaddr_t *sent_ip;
static const uint8_t *sent_data;
static int intg = 0;
static int frac = 0;

//Used to sum the autocorrelation
static float real;
static float img;

/*-------------------------------------------------------------------------*/
//This defines the process used upon startup
PROCESS(client_process, "Sensor node client process");
AUTOSTART_PROCESSES(&client_process);
/*-------------------------------------------------------------------------*/
/*
 * There is no native math library for Contiki so here is a 3 term
 * series approximation of arctangent.
 *
 * Input: n - numerator, d - denominator
 * Output: Arctangent
 */
static float
atanSeries(float n, float d) {

  float r = n/d;

  if(r>1 || r <-1) {
    r = (d/n);
    return -90 - 57.3*(r-(r*r*r/3));
  } else {
    return 57.3*(r-(r*r*r/3));
  }

}
/*-------------------------------------------------------------------------*/
/*
 * This method is the callback associated with the backoff_timer and is used
 * to reduce collisions when responding to a beacon
 */
static void
send(void* in)
{
  static char buf[20];
```

```
      sprintf(buf, "%d.%d %s", intg, ABS(frac), sent_data);
      simple_udp_sendto(&unicast_connection, buf, strlen(buf) + 1, sent_ip);


}
/*-------------------------------------------------------------------------*/
/*
 * This method is called in response to an incoming UDP packet from the 4321
 * port meant to receive beacons. It calculate the Doppler velocity and sets
 * a backoff timer.
 */
static void
receiver(struct simple_udp_connection *c,
         const uip_ipaddr_t *sender_addr,
         uint16_t sender_port,
         const uip_ipaddr_t *receiver_addr,
         uint16_t receiver_port,
         const uint8_t *data,
         uint16_t datalen)
{

   //calculate the Doppler velocity
   float vel = atanSeries(img,real)*(5.2/2);
   intg = (int)(vel);
   frac = (int)((vel-intg)*1000);
   real = 0;
   img = 0;

   //remeber the beacons IP address and the beacon number
   sent_ip = sender_addr;
   sent_data = data;

   //set the backoff_timer
   ctimer_set(&backoff_timer, BACKOFF_TIME, send, NULL);


}
/*-------------------------------------------------------------------------*/
/*
 * This is the main process thread and it initializes the radar, initializes
 * UDP port connections, and contains a paced loop to read the radar IQ pairs
 * from the ADC.
 */
PROCESS_THREAD(client_process, ev, data)
{
   //the periodic timer is used to regulate when the radar is read
   static struct etimer periodic;

   //this structure hold the last complex IQ pair radar reading
   static struct cplx_num prev;

   PROCESS_BEGIN();

   //Make a call to the radar drives to initialize them
   SENSORS_ACTIVATE(radar_sensor);

   //setup both UDP connections
   simple_udp_register(&unicast_connection, UDP_PORT,
                       NULL, UDP_PORT, NULL);

   simple_udp_register(&broadcast_connection, UDP_PORT_BROAD,
                       NULL, UDP_PORT_BROAD,
                       receiver);

   //set the initial values for the autocorrelation summation
   radar_sensor.value(0);
   prev.real = radar_value.real;
   prev.img = radar_value.img;
```

A-3

```
  real = 0;
  img = 0;

  //set the periodic timer to 500Hz
  etimer_set(&periodic, CLOCK_SECOND/500);
  while(1) {
    //allows multitasking
    PROCESS_YIELD();

    //only read radar when the periodic timer has expired
    if(etimer_expired(&periodic)) {
      //reset the periodic timer so that it really is periodic
      etimer_reset(&periodic);

      //make a call to the radar drivers to read the IQ pair
      radar_sensor.value(0);

      //calculate the next step in the autocorrelation summation
      //sum real and imaginary parts of Z(N-1)*Z(N)
      real += ((radar_value.real*(float)(prev.real)) -
              ((radar_value.img*-1)*(float)(prev.img)));
      img  += ((radar_value.real*(float)(prev.img))  +
              ((radar_value.img*-1)*(float)(prev.real)));

      //set previous value for next calculation
      prev.real = radar_value.real;
      prev.img = radar_value.img;

    }
  }

  SENSORS_DEACTIVATE(radar_sensor);

  PROCESS_END();
}
/*-------------------------------------------------------------------------*/
```

host.c

```c
/*
 * host.c
 * Michael K. Blaser
 * Fall 2011
 *
 * This program is meant to be run on the sink node.
 */

/*
 * Contiki OS BSD License
 *
 * Copyright (c) 2011, Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the Institute nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */

#include "contiki.h"
#include "lib/random.h"
#include "sys/ctimer.h"

#include "net/uip.h"
#include "net/uip-debug.h"
#include "net/netstack.h"

#include "simple-udp.h"

#include <stdio.h>
#include <string.h>

//This port is used to receive messages from the sensor nodes
#define UDP_PORT 1234
//This port is used to send beacon messages to the sensor nodes
#define UDP_PORT_BROAD 4321

//This sets the poll rate of the beacon (2Hz)
#define SEND_INTERVAL   (0.5 * CLOCK_SECOND)

static struct simple_udp_connection broadcast_connection;
static struct simple_udp_connection unicast_connection;
```

```c
static struct ctimer broadcast_timer;

/*-------------------------------------------------------------------------*/
//This defines the process used upon startup
PROCESS(host_process, "Sink node host process");
AUTOSTART_PROCESSES(&host_process);
/*-------------------------------------------------------------------------*/
/*
 * This method is called in response to an incoming UDP packet from the 1234
 * port meant to receive sensor data. It prints the data to the serial link
 * to the host PC.
 */
static void
receiver(struct simple_udp_connection *c,
         const uip_ipaddr_t *sender_addr,
         uint16_t sender_port,
         const uip_ipaddr_t *receiver_addr,
         uint16_t receiver_port,
         const uint8_t *data,
         uint16_t datalen)
{
  printf("%s %d\n",data,sender_addr->u8[sizeof(sender_addr->u8)-1]);
}
/*-------------------------------------------------------------------------*/
/*
 * This method is the callback associated to the broadcast timer. Each beacon
 * contains a beacon message number so the host PC can match beacon responses
 * from the sensor data.
 */
static void
broadcast(void* in) {
  static unsigned int message_number;
  static char buf[20];
  static uip_ipaddr_t addr;

  //Reset the callback timer so it is periodic
  ctimer_reset(&broadcast_timer);

  //Create the beacon message with the beacon message number
  sprintf(buf, "%d", message_number);
  message_number++;

  //Set send address to be a broadcast message
  uip_create_linklocal_allnodes_mcast(&addr);

  //Send the beacon message
  simple_udp_sendto(&broadcast_connection, buf, strlen(buf) + 1, &addr);
}
/*-------------------------------------------------------------------------*/
/*
 * This is the main process thread and it set the radio duty cycle to 100%,
 * initializes the UDP port connections, and sets a periodic callback timer for
 * the beacon.
 */
PROCESS_THREAD(host_process, ev, data)
{
  PROCESS_BEGIN();

  //The data sink runs with a 100% duty cycle in order to ensure high
  //packet reception rates.
  NETSTACK_MAC.off(1);

  simple_udp_register(&unicast_connection, UDP_PORT,
                      NULL, UDP_PORT, receiver);

  simple_udp_register(&broadcast_connection, UDP_PORT_BROAD,
                      NULL, UDP_PORT_BROAD, NULL);
```

A-6

```
  //Sets the callback timer for the beacon
  ctimer_set(&broadcast_timer, SEND_INTERVAL, broadcast, NULL);

  while(1) {
    //allows multitasking
    PROCESS_WAIT_EVENT();
  }

  PROCESS_END();
}
/*-------------------------------------------------------------------------*/
```

radar-sensor.h

```c
/*
 * radar-sensor.h
 * Michael K. Blaser
 * Fall 2011
 *
 * This program provides the radar sensor driver interface for the sensor
nodes.
 */

#ifndef __RADAR_SENSOR_H__
#define __RADAR_SENSOR_H__

//Use the Contiki sensor code base
#include "lib/sensors.h"

//This structure is used to store an IQ pair from the ADC
struct cplx_num {
  int16_t real;
  int16_t img;
} radar_value;

//This allows for the creation of the radar sensor through the Contiki OS
extern const struct sensors_sensor radar_sensor;

#define RADAR_SENSOR "Radar"

#endif /* __RADAR_SENSOR_H__ */
```

radar-sensor.c

```c
/*
 * radar-sensor.c
 * Michael K. Blaser
 * Fall 2011
 *
 * This program provides the radar sensor drivers used by the sensor nodes.
 */

#include "dev/radar-sensor.h"
#include "dev/sky-sensors.h"

#include "contiki.h"

//These includes allow for low level access to the MSP430 microcontroller
#ifdef __IAR_SYSTEMS_ICC__
#include <msp430.h>
#else
#include <io.h>
#include <signal.h>
#endif


// Configure ADC12_2 to repeatedly sample channel 4 and 5 (voltage)
#define INPUT_CHANNEL           ((1 << INCH_2) | (1 << INCH_4))
#define INPUT_REFERENCE         SREF_0

#define RADAR_MEM1              ADC12MEM2
#define RADAR_MEM2              ADC12MEM4

//This allows for the creation of the radar sensor through the Contiki OS
const struct sensors_sensor radar_sensor;

/*---------------------------------------------------------------------------*/
/*
 * This method takes the most recent ADC readings and stores them into the IQ
 * pair structure "radar_value".
 */
static int
value(int type)
{
  radar_value.real = RADAR_MEM2-2252; //take off 1.65V offset from radar
  radar_value.img = RADAR_MEM1-2252;
  return 1;
}
/*---------------------------------------------------------------------------*/
/*
 * This method allows the Contiki OS to configure the sensor.
 */
static int
configure(int type, int c)
{
  return sky_sensors_configure(INPUT_CHANNEL, INPUT_REFERENCE, type, c);
}
/*---------------------------------------------------------------------------*/
/*
 * This method allows the Contiki OS to read the status of the sensor.
 */
static int
status(int type)
{
  return sky_sensors_status(INPUT_CHANNEL, type);
}
/*---------------------------------------------------------------------------*/
```

```
//This sets up how the sensor appears to the Contiki OS
SENSORS_SENSOR(radar_sensor, RADAR_SENSOR, value, configure, status);
```

tracker.m

```
% tracker.m
% Michael K. Blaser
% Fall 2011
%
% This program runs on the host PC.

%These vectors contain the sensor node locations found by manual measurement
SX = [-91.44;91.44];
SY = [0.01;0.01];

%This is the inital state of the EKF and depends on the test setup
xhatold = [0;75;0;0];
Pold = zeros(4,4);

%Used to create the serial link to the sink node
s = serial('COM17');
set(s,'BaudRate',115200,'DataBits',8);
fopen(s)

%setup some inital values
laststep = 0;
msgcount = 0;
z = zeros(2,1);

while (1)

    %read the serial link line by line
    line = fgetl(s);
    %if there is an error reading a line stop tracking
    if(line == -1)
        break;
    end

    %parse and match sensor results
    %msg: F.F NUM ID
    reading = sscanf(line,'%f%d%d');

    %Grab measurment and threshold at 5cm/s
    meas = reading(1);
    if(abs(meas) < 5)
        meas = 0;
    end

    %Grab the beacon message number and sensor ID
    msgnum = reading(2);
    id = reading(3);

    %Accept only newer readings
    if(msgcount <= msgnum)
        %store reading properly in sensor array by sensor ID
        if (id == 153)
            z(1) = meas;
        elseif (id == 229)
            z(2) = meas;
        else
            'ERROR: not a matching address'
        end

        %if messages match than send to EKF
        if (msgcount == msgnum)
```

A-10

```
            %set timestep depending on last matched beacon
            if(laststep == 0)
                t = .5;
            else
                t = .5*(msgcount-laststep);
            end
            %store timestep to determine future timestep
            laststep = msgcount;

            %Calculate an iteration of the EKF
            [xhat,P] = EKF2(xhatold,Pold,z,t,SX,SY);

            %display results
            fprintf('%f\t%f\t%f\t%f\n',xhat(1),xhat(2),xhat(3),xhat(4));

            %store results for next iteration
            xhatold = xhat;
            Pold = P;
        end

        %set the beacon message number for beacon matching
        msgcount = msgnum;
    end
end

%Used to end serial link
fclose(s)
delete(s)
clear s

%This method calculates and iteration of the EKF.
% input:
%       xhatold - previous iteration state
%       Pold - previous iteration state covariance
%       z - measurement vector
%       t - timestep
%       SX - vector of sensor x locations
%       SY - vector of sensor y locations
% output:
%        xhat - calculated state
%        P - calculated covarience
function [xhat,P] = EKF2(xhatold,Pold,z,t,SX,SY)
%State trasition model matrix A
A = [1,0,t,0;0,1,0,t;0,0,1,0;0,0,0,1];
%State trasition model noise matrix
R = (2/5).*eye(2);
%Observation model noise matrix
Q = [.5,0,t,0;0,.5,0,t;0,0,.5,0;0,0,0,.5];

% Time update
xhatminus = A*xhatold;
Pminus = A*Pold*A'+Q;

%Calculate Jacobian matrix
J = findJ(xhatminus,SX,SY);

% Measurement update
K = (Pminus*(J'))/(J*Pminus*(J')+R);
xhat = xhatminus+K*(z-F(xhatminus,SX,SY));
P = (eye(4)-K*J)*Pminus;

%This method calculates Jacobian matrix for the measurement update.
% input:
%       x - time step state
%       SX - vector of sensor x locations
%       SY - vector of sensor y locations
% output:
```

```
%          J - Jacobian matrix
function [J] = findJ(x,SX,SY)
J = zeros(2,4);
for i = 1:2
  J(i,1) = (x(3)*(SX(i)-x(1))*(SY(i)-x(2))-x(4)*(SX(i)-x(1))^2)/
           (((SX(i)-x(1))^2+(SY(i)-x(2))^2)^(3/2));
  J(i,2) = (x(4)*(SX(i)-x(1))*(SY(i)-x(2))-x(3)*(SY(i)-x(2))^2)/
           (((SX(i)-x(1))^2+(SY(i)-x(2))^2)^(3/2));
  J(i,3) = (SX(i)-x(1))/sqrt(((SX(i)-x(1))^2+(SY(i)-x(2))^2));
  J(i,4) = (SY(i)-x(2))/sqrt(((SX(i)-x(1))^2+(SY(i)-x(2))^2));
end

%This method calculates the expected measurements vector.
% input:
%        x - time step state
%        SX - vector of sensor x locations
%        SY - vector of sensor y locations
% output:
%         out - expected measurement vector
function [out] = F(x,SX,SY)
out = zeros(2,1);
for i = 1:2
  out(i) = (x(3)*(SX(i)-x(1))+x(4)*(SY(i)-x(2)))/
           sqrt((SX(i)-x(1))^2+(SY(i)-x(2))^2);
end
```

A-12

Appendix B Datasets

Perpendicular

Trial 1

-1. 350 218 229	-3. 264 229 153	-8. 314 240 153	12. 843 253 153
0. 924 218 153	-1. 383 229 229	0. 768 240 229	3. 177 253 229
-2. 259 219 153	-3. 300 230 153	-5. 594 241 153	8. 76 254 153
0. 158 219 229	-5. 370 230 229	0. 503 241 229	2. 297 254 229
0. 37 220 153	-4. 232 231 153	-7. 186 243 153	1. 781 255 153
-2. 195 220 229	-8. 232 231 229	-2. 960 243 229	4. 842 255 229
0. 370 221 153	-4. 604 232 153	4. 148 244 153	4. 832 256 229
-1. 215 221 229	-5. 15 232 229	3. 190 244 229	3. 213 256 153
2. 439 222 153	-4. 516 233 229	8. 172 245 153	6. 746 257 153
0. 811 222 229	-9. 200 233 153	5. 681 245 229	3. 440 257 229
0. 393 223 229	-5. 98 234 153	10. 509 246 153	4. 242 259 153
-1. 793 223 153	-4. 884 234 229	4. 472 246 229	1. 420 259 229
-4. 476 224 153	-4. 453 235 153	10. 780 247 153	0. 36 260 153
0. 232 224 229	-2. 123 235 229	7. 414 247 229	0. 362 260 229
-7. 923 225 153	-6. 327 236 153	5. 578 248 153	-1. 319 261 229
0. 956 225 229	-3. 640 236 229	9. 789 248 229	2. 370 261 153
-5. 91 226 153	-7. 365 237 153	7. 99 250 229	1. 65 262 229
-5. 755 226 229	-5. 853 237 229	11. 362 250 153	7. 910 262 153
-2. 727 227 229	-3. 262 238 229	3. 636 251 153	0. 953 263 229
-4. 877 227 153	-7. 39 238 153	6. 587 251 229	3. 225 263 153
-4. 141 228 229	-7. 306 239 153	6. 689 252 153	-2. 680 264 153
-5. 605 228 153	-2. 913 239 229	3. 380 252 229	

Trial 2

0. 897 264 229	-2. 454 280 153	-3. 783 290 229	-5. 680 300 153
0. 763 265 153	0. 336 281 153	-2. 977 291 229	-1. 131 299 229
0. 581 265 229	0. 600 280 229	-4. 803 291 153	0. 223 300 229
0. 913 266 153	-3. 124 283 153	-4. 469 292 153	-9. 150 301 153
0. 408 266 229	0. 65 281 229	-8. 250 292 229	-4. 796 302 153
-2. 683 267 153	-2. 977 282 229	-3. 85 293 153	-4. 465 301 229
0. 105 267 229	-4. 923 284 153	-3. 322 293 229	-5. 617 302 229
0. 133 268 229	-6. 679 285 153	-5. 519 294 153	-5. 611 303 153
-1. 191 268 153	0. 0 283 229	-2. 628 294 229	0. 963 303 229
0. 213 269 153	1. 849 284 229	-10. 71 295 153	-6. 204 304 153
0. 485 269 229	-1. 292 286 153	-3. 732 295 229	-3. 267 304 229
0. 318 270 153	0. 28 285 229	-5. 429 296 229	-7. 176 305 153
0. 252 271 153	-1. 222 286 229	-5. 402 296 153	-4. 528 305 229
-1. 191 270 229	0. 516 287 153	-5. 11 297 153	-4. 811 306 153
0. 57 272 153	-4. 653 287 229	-8. 724 297 229	-2. 955 306 229
0. 302 271 229	-1. 442 288 153	-3. 472 298 153	-5. 79 307 153
0. 150 272 229	-7. 931 288 229	-11. 92 299 153	-3. 337 307 229
0. 457 273 153	-5. 477 290 153	-3. 395 298 229	0. 513 308 229

```
-6. 316  308  153          5. 502  315  229          8. 953  322  153          0. 29  327  229
0. 133  309  153           6. 379  316  153          2. 763  321  229          1. 102  328  153
1. 81  309  229            11. 525  317  153         1. 361  322  229          0. 29  328  229
3. 852  310  153           4. 319  316  229          3. 206  323  153          0. 0  329  229
6. 911  310  229           9. 310  318  153          3. 808  323  229          1. 821  329  153
5. 548  311  153           8. 308  317  229          2. 598  324  153          5. 757  330  153
6. 273  311  229           9. 264  318  229          3. 900  324  229          0. 749  330  229
6. 729  312  153           7. 316  319  153          2. 599  325  229          0. 563  331  229
4. 343  312  229           8. 328  319  229          8. 312  325  153          4. 106  331  153
4. 767  314  229           6. 416  320  153          11. 1  326  153           0. 210  332  229
7. 656  314  153           6. 691  321  153          3. 16  326  229           -1. 101  332  153
6. 926  315  153           9. 578  320  229          8. 115  327  153          0. 432  333  229
```

Trial 3

```
-2. 918  333  153          -1. 4  353  153           -5. 764  370  153         3. 419  385  229
0. 100  334  229           -1. 674  354  153         -1. 86  370  229          6. 121  386  153
0. 551  334  153           -4. 140  354  229         -4. 753  371  153         5. 106  386  229
-2. 600  335  153          -5. 836  355  153         -2. 663  371  229         10. 296  387  229
0. 662  335  229           0. 507  355  229          -7. 191  372  153         7. 71  387  153
0. 474  336  153           -6. 827  356  153         -2. 457  372  229         8. 734  388  153
0. 821  336  229           -2. 287  356  229         -7. 481  373  153         2. 213  388  229
0. 504  337  153           -8. 261  357  229         -4. 693  373  229         6. 108  389  153
0. 259  337  229           -6. 807  357  153         -3. 213  374  153         0. 542  389  229
0. 21  338  153            -1. 27  358  153          -3. 496  374  229         0. 146  390  229
0. 75  338  229            -6. 996  358  229         -5. 72  375  153          4. 59  390  153
0. 185  339  229           -6. 610  359  229         1. 689  376  153          5. 37  391  153
0. 698  339  153           -1. 754  359  153         0. 299  375  229          0. 133  391  229
0. 140  340  229           -4. 138  360  153         7. 213  377  153          0. 0  392  229
0. 404  340  153           -3. 551  360  229         0. 746  376  229          10. 100  392  153
-1. 388  341  153          -4. 732  361  229         7. 913  377  229          7. 946  393  153
0. 432  341  229           -4. 911  361  153         3. 847  378  229          0. 0  393  229
2. 902  342  153           -2. 671  362  229         8. 566  378  153          3. 343  394  153
3. 508  347  153           -8. 576  362  153         10. 759  379  153         0. 376  394  229
0. 334  347  229           -3. 334  363  229         5. 877  379  229          2. 203  395  153
-1. 700  348  153          -8. 405  363  153         8. 642  380  229          0. 345  395  229
0. 713  348  229           -4. 580  364  229         12. 111  380  153         0. 131  396  229
0. 958  349  153           -4. 124  364  153         8. 226  381  153          0. 680  396  153
-1. 73  349  229           -4. 531  365  153         11. 298  381  229         0. 148  397  229
-3. 780  350  153          -3. 189  365  229         6. 344  382  153          6. 433  397  153
1. 575  350  229           -2. 924  366  229         5. 83  382  229           5. 428  398  153
0. 184  351  229           -5. 152  366  153         7. 85  383  153           0. 960  398  229
-4. 575  351  153          -9. 398  367  153         6. 231  383  229          -4. 75  399  153
-2. 59  352  229           -7. 34  367  229          5. 598  384  229
-2. 123  352  153          -9. 216  368  153         9. 276  384  153
-1. 132  353  229          -3. 746  368  229         9. 305  385  153
```

Trial 4

```
-2. 282  400  153          0. 420  407  153          -3. 710  420  153         -6. 54  428  153
0. 851  399  229           0. 502  408  229          -1. 187  420  229         -4. 415  427  229
0. 35  400  229            0. 207  408  153          -8. 286  421  153         -6. 707  428  229
0. 675  401  153           0. 880  414  153          -2. 49  421  229          -2. 993  429  153
0. 0  401  229             0. 492  414  229          -3. 401  422  229         -2. 281  430  153
0. 630  402  229           -1. 671  415  153         -4. 995  422  153         -2. 867  429  229
0. 609  403  153           0. 521  415  229          -6. 109  423  229         -5. 196  431  153
0. 976  403  229           -4. 160  416  153         -2. 136  423  153         -5. 549  430  229
0. 383  404  153           2. 994  416  229          0. 152  424  153          -2. 994  431  229
0. 236  404  229           1. 705  417  153          0. 744  424  229          -7. 934  432  153
0. 517  405  153           -1. 955  417  229         -6. 252  425  153         -6. 152  433  153
0. 766  405  229           0. 691  418  229          -2. 85  425  229          -6. 81  432  229
1. 132  406  229           1. 989  418  153          -5. 490  426  153         -5. 782  433  229
0. 633  406  153           0. 743  419  229          -6. 63  427  153          -3. 472  434  153
0. 257  407  229           -2. 291  419  153         -1. 76  426  229          -1. 365  434  229
```

```
-6.454  436  153      -2.66  443  229       5.224  452  229       0.906  460  229
-3.773  436  229       4.613  444  229       5.427  453  229       0.292  460  153
-8.86  437  153        7.855  445  153      13.787  453  153       1.767  461  153
-5.226  437  229       5.304  445  229       4.82  454  153        3.948  461  229
-10.991  438  153     10.652  446  153       0.840  454  229       5.377  462  153
-1.656  438  229       5.813  446  229       4.334  455  153       0.715  462  229
-2.726  439  229       9.380  448  153       4.994  455  229       7.80  463  153
-5.901  439  153       6.917  448  229       2.574  456  153       0.774  463  229
-3.936  440  229       9.485  449  229       8.227  456  229       0.756  464  229
-2.177  440  153       9.979  449  153       2.883  457  153       3.953  464  153
-6.408  441  153       9.822  450  153       2.903  457  229       0.315  465  153
-3.904  441  229       0.207  450  229       7.597  458  153       0.793  465  229
-5.817  442  153       5.513  451  153       2.645  458  229       0.278  466  153
-3.340  442  229       0.746  451  229       5.794  459  153       0.0  466  229
-5.233  444  153       5.736  452  153       0.212  459  229
```

Trial 5

```
-1.496  467  153      -2.101  485  229      -3.543  501  229       8.134  517  153
0.743  467  229       -2.406  486  229      -6.235  502  153       5.889  517  229
0.620  468  153        0.152  486  153      -4.989  502  229      11.58  518  229
0.363  468  229       -3.530  487  153      -7.306  503  153       5.605  518  153
-2.71  469  153       -4.74  487  229       -4.828  503  229       6.93  519  153
0.633  469  229       -6.973  488  153      -4.542  504  153       3.944  519  229
0.160  470  229        0.735  488  229      -2.859  504  229       7.50  520  153
0.537  470  153       -4.301  490  229      -6.811  505  229       3.330  520  229
-1.398  471  153      -3.446  490  153      -6.41  505  153        5.561  521  153
0.0  471  229         -3.503  491  153       0.731  506  229       0.379  521  229
1.157  472  153       -2.788  491  229      -2.759  506  153       1.144  522  229
0.547  472  229       -2.217  492  229       3.521  507  153       0.744  523  153
0.587  473  229       -11.86  492  153       2.2  507  229         1.65  523  229
0.391  473  153       -8.311  493  229      10.72  508  153        3.628  524  153
0.754  474  229       -8.931  493  153       8.995  508  229       2.325  524  229
1.173  474  153       -6.577  494  153       4.0  509  153         7.532  525  153
0.843  475  229       -5.583  494  229       8.218  509  229       0.755  525  229
0.929  475  153       -3.321  495  153       9.136  510  153       0.0  526  229
-1.461  478  229      -10.574  495  229      8.713  510  229       4.643  526  153
1.150  479  153       -5.872  496  153      11.956  511  153       2.811  527  153
-1.247  480  153      -1.504  496  229       4.94  511  229        0.69  527  229
-3.371  479  229      -5.676  497  153       6.246  512  153       0.365  528  229
0.328  480  229       -2.937  497  229       4.800  512  229       0.864  528  153
0.597  482  153       -5.616  498  229       5.15  513  153        0.322  529  229
0.262  482  229       -10.230  498  153      4.304  513  229       2.304  529  153
-8.438  483  153      -4.774  499  229       4.365  514  153       5.504  530  153
0.24  483  229        -5.608  499  153       4.403  514  229      -1.234  530  229
-4.321  484  153      -2.731  500  229       7.971  515  229
-2.345  484  229      -2.110  500  153       5.65  516  229
-1.340  485  153      -10.633  501  153      9.845  516  153
```

Oblique

Trial 1

```
1.177  531  153        0.397  537  153       0.518  565  153      -4.89  571  153
-1.225  532  153       0.740  537  229      -1.428  565  229      -1.314  571  229
0.691  531  229        0.72  538  153        0.744  566  153      -6.738  572  153
0.132  532  229        0.395  538  229       0.132  566  229      -4.993  572  229
0.651  533  153        0.321  539  153      -2.989  567  153      -9.112  573  153
0.270  533  229        0.128  539  229       0.851  567  229      -2.254  573  229
-6.110  534  153       0.890  561  153      -1.106  568  153       0.947  574  229
0.584  534  229       -1.882  561  229      -4.852  568  229      -1.684  574  153
-4.576  535  153       0.574  562  153      -1.886  569  229      -2.114  575  229
0.695  535  229       -3.243  562  229      -2.687  569  153      -5.880  575  153
0.965  536  153       -5.414  563  229      -2.412  570  153      -3.97  576  229
0.96  536  229        -1.104  563  153      -2.367  570  229      -4.468  576  153
```

```
-5.159 577 153      6.332 581 229      4.760 587 229      1.248 591 229
0.832 577 229       4.239 582 153      2.985 587 153      1.193 592 153
4.611 578 229       5.738 582 229      2.422 588 229      0.230 593 153
3.936 578 153       7.122 583 153      2.861 588 153      5.881 592 229
5.780 579 153       2.541 583 229      0.405 589 153      1.821 593 229
3.538 579 229       2.818 585 153      2.227 589 229      -1.391 594 153
0.65 580 229        1.235 585 229      -1.291 590 153     1.124 594 229
4.281 580 153       0.342 586 153      2.338 590 229      -1.884 595 153
2.140 581 153       5.273 586 229      -2.544 591 153     0.11 595 229
```

Trial 2

```
0.684 596 153       -1.204 601 153     -5.755 608 229     0.507 614 229
0.0 596 229         0.623 601 229      -1.201 608 153     0.981 614 153
0.708 597 229       0.990 602 153      -1.172 610 229     3.596 615 153
0.510 597 153       -1.536 602 229     0.637 610 153      0.872 615 229
0.251 598 229       0.840 603 153      0.142 611 229      -2.72 616 229
1.79 598 153        1.195 603 229      -2.402 611 153     0.155 616 153
0.119 599 229       0.898 604 153      -4.915 612 229     -5.887 617 229
-1.104 599 153      0.687 604 229      0.219 612 153      -4.476 617 153
0.138 600 153       0.750 607 229      0.822 613 229
0.257 600 229       2.63 607 153       0.700 613 153
```

Trial 3

```
-2.251 618 153      2.58 660 153       -3.750 671 229     1.415 684 153
0.24 619 229        -6.160 660 229     -4.429 672 153     2.149 685 229
3.438 619 153       0.19 661 229       0.373 673 153      1.940 685 153
0.195 620 153       3.216 661 153      0.735 672 229      1.514 686 229
0.17 620 229        2.813 662 153      -3.703 674 153     1.90 686 153
-1.425 621 229      0.871 662 229      -3.620 673 229     8.135 688 229
-1.311 621 153      1.618 663 153      -2.182 675 153     0.752 688 153
-4.8 622 153        -1.825 663 229     -9.528 676 153     0.388 689 153
-6.57 622 229       1.492 664 153      -1.500 674 229     5.467 689 229
-11.625 623 153     -2.680 664 229     -2.380 675 229     0.938 690 229
-6.507 623 229      -3.365 665 153     -3.725 677 153     1.106 690 153
-3.878 624 153      0.630 665 229      2.804 678 153      4.992 691 229
0.749 624 229       -2.258 666 229     8.45 679 153       0.632 691 153
-3.229 625 153      -2.245 666 153     -5.602 676 229     4.813 692 229
-1.432 625 229      -2.596 667 229     5.645 681 153      -4.274 692 153
-6.220 627 153      -2.774 667 153     -4.246 677 229     1.755 693 229
-2.180 627 229      0.855 668 153      2.64 679 229       -2.90 693 153
0.545 657 229       0.716 668 229      4.44 681 229       1.735 694 229
0.788 657 153       -5.213 669 229     3.349 682 153      0.230 694 153
0.55 658 153        0.657 669 153      1.951 682 229      2.419 695 153
-3.9 658 229        -3.801 670 229     2.235 683 153      4.14 695 229
0.282 659 229       -3.433 670 153     4.926 683 229
0.525 659 153       -9.617 671 153     1.312 684 229
```

Trial 4

```
0.561 719 229       0.373 722 229      -3.856 740 153     -2.7 744 229
0.257 730 229       0.595 734 153      0.623 740 229      -2.474 745 229
0.914 730 153       0.141 734 229      -1.931 741 153     1.749 745 153
0.282 731 153       0.516 731 229      -2.422 741 229     -1.448 746 153
0.589 720 229       0.317 735 153      0.906 742 153      0.711 746 229
0.92 732 229        1.101 733 229      0.351 742 229      2.46 747 153
0.529 732 153       0.325 735 229      2.111 743 153      0.751 747 229
0.348 721 229       0.451 736 153      1.192 743 229      0.248 748 229
0.493 733 153       0.15 736 229       0.19 744 153       0.553 748 153
```

```
-5. 304  749  153      -5. 556  756  153      5. 839  763  229      6. 741  770  229
-3. 789  750  153      -1. 420  756  229      2. 171  764  153      5. 464  771  229
-1. 31  749  229       -6. 479  757  153      3. 206  764  229      -2. 847  771  153
0. 631  751  153       -2. 196  758  153      0. 742  765  153      -1. 407  772  153
-1. 992  750  229      0. 391  757  229       0. 464  765  229      1. 460  772  229
0. 657  752  153       -2. 429  758  229      0. 52  766  153       0. 945  773  153
-3. 95  751  229       -1. 689  759  153      2. 291  766  229      3. 177  773  229
-2. 710  753  153      0. 224  759  229       8. 434  767  229      0. 327  774  153
0. 380  752  229       2. 752  760  153       0. 25  767  153       6. 269  774  229
0. 315  753  229       6. 671  760  229       5. 46  768  153       -2. 285  775  153
-3. 394  754  153      2. 646  761  153       6. 916  768  229      3. 666  775  229
-5. 177  754  229      5. 622  761  229       6. 575  769  229      0. 878  776  229
-9. 767  755  153      5. 698  762  229       1. 796  769  153
-6. 689  755  229      10. 164  763  153      -2. 42  770  153
```

Trial 5

```
-4. 1  776  153        0. 318  788  153       0. 509  800  153      0. 211  811  153
1. 294  777  229       1. 353  789  153       -4. 346  801  229     5. 324  812  229
0. 390  777  153       -2. 301  789  229      -6. 542  801  153     2. 251  812  153
0. 686  778  153       0. 464  790  153       -5. 264  802  153     3. 99  813  229
0. 85  778  229        -1. 300  790  229      -2. 781  802  229     4. 15  813  153
1. 562  779  153       0. 938  791  153       -3. 865  803  229     -1. 3  814  153
-1. 23  779  229       -5. 450  791  229      -5. 989  803  153     2. 84  814  229
1. 519  780  229       0. 535  792  153       -1. 23  804  153      0. 498  815  153
0. 684  780  153       -3. 644  792  229      -6. 393  804  229     9. 227  815  229
0. 602  781  153       -3. 508  793  229      0. 814  805  153      6. 965  816  229
0. 26  781  229        1. 700  793  153       0. 332  805  229      0. 903  816  153
1. 192  782  229       0. 630  794  153       -5. 947  806  153     2. 681  817  229
0. 182  782  153       -2. 433  794  229      -6. 705  806  229     3. 65  817  153
0. 834  783  153       -1. 398  795  153      -3. 94  807  153      0. 311  818  153
0. 121  783  229       0. 749  795  229       0. 392  807  229      5. 92  818  229
0. 359  785  153       -4. 176  796  229      7. 690  808  153      -4. 559  819  153
0. 55  785  229        0. 547  796  153       7. 458  808  229      -2. 222  820  153
1. 137  786  153       0. 767  797  229       3. 491  809  229      4. 542  819  229
0. 842  786  229       -2. 396  798  229      3. 139  809  153      0. 238  821  153
-2. 353  787  229      -5. 28  797  153       6. 350  810  229      4. 519  820  229
0. 808  787  153       0. 55  798  153        2. 547  810  153      3. 288  822  153
-3. 864  788  229      -1. 20  800  229       6. 43  811  229       5. 506  821  229
```

Diamond

Trial 1

```
0. 36  1049  229       0. 78  1057  153       -8. 141  1064  229    0. 996  1073  153
0. 299  1049  153      0. 0  1057  229        -4. 312  1066  153    10. 433  1073  229
-2. 841  1050  153     0. 394  1058  153      -1. 974  1065  229    1. 954  1074  153
0. 337  1050  229      0. 540  1058  229      0. 116  1067  153     6. 487  1074  229
-1. 306  1051  153     -7. 865  1059  229     -6. 430  1066  229    2. 308  1075  153
-1. 704  1052  153     5. 302  1059  153      -1. 108  1068  153    1. 688  1075  229
0. 925  1053  153      -2. 209  1060  229     1. 49  1067  229      10. 652  1076  153
0. 384  1051  229      2. 507  1060  153      -11. 249  1069  153   8. 481  1076  229
-1. 389  1054  153     -4. 78  1061  229      -3. 952  1068  229    14. 478  1077  153
0. 229  1052  229      -1. 660  1061  153     -7. 94  1069  229     8. 187  1077  229
0. 241  1053  229      0. 425  1062  153      -9. 843  1070  153    9. 508  1078  153
0. 523  1055  153      -3. 265  1062  229     0. 0  1070  229       0. 260  1078  229
0. 480  1054  229      -2. 175  1063  153     2. 303  1071  153     0. 0  1079  229
0. 787  1056  153      -1. 196  1063  229     7. 160  1071  229     18. 460  1079  153
0. 542  1055  229      -8. 115  1064  153     9. 827  1072  229     0. 0  1080  229
0. 0  1056  229        -4. 728  1065  153     4. 503  1072  153     5. 537  1080  153
```

Trial 2

```
-2.594  864  153        -1.190  874  153        -6.100  898  153         1.918  906  153
 0.741  864  229         0.440  875  153        -1.623  897  229         8.111  906  229
-1.643  865  153         0.192  876  153        -9.500  899  153         3.874  907  153
 0.774  865  229         0.111  891  229        -5.66  898  229          6.287  907  229
 0.28  868  153          0.295  892  153        -3.902  900  153         5.969  908  229
 0.187  868  229         4.59  892  229          0.632  899  229         3.440  909  229
 0.457  869  153         5.821  893  153        -6.756  901  153         5.460  908  153
 0.257  870  229        -1.368  893  229        -4.189  900  229         8.196  909  153
 0.957  870  153        -4.424  894  229        -7.130  901  229         7.56  910  229
 0.137  871  153         0.247  894  153        -5.344  903  229         0.447  911  229
 0.279  871  229        -1.665  895  153        -10.597  903  153        14.238  910  153
-1.95  872  153         -5.447  895  229         5.29  904  153          16.392  911  153
 0.663  872  229        -4.375  896  153         16.36  904  229         0.113  912  229
 0.607  873  153         0.905  897  153         2.792  905  229         0.226  913  229
 0.154  873  229        -5.522  896  229         4.781  905  153
```

Trial 3

```
 1.393  923  153        -7.42  931  153         -6.177  961  153         0.376  970  153
 0.0  921  229           0.516  954  153         0.837  962  229         15.128  971  229
 0.0  922  229           0.866  954  229        -6.323  962  153        -1.189  971  153
 2.474  924  153         3.847  955  153        -7.856  963  153         5.282  972  153
 0.169  923  229         0.850  955  229         0.751  963  229         9.723  972  229
 2.757  925  153         0.392  956  229         0.186  964  229         9.576  973  153
-2.806  924  229         7.95  956  153         -4.849  964  153         4.773  973  229
-2.138  925  229         1.689  957  153        -2.240  965  153         15.215  974  153
-2.237  926  229         0.522  957  229         0.635  965  229         0.802  974  229
-2.886  926  153        -2.848  958  153        -15.324  966  153        5.394  975  153
-2.265  927  153         0.513  958  229         2.191  966  229         0.753  975  229
-4.757  927  229        -1.312  959  153        -4.277  967  153         7.656  976  153
-3.152  929  229        -3.962  960  153         6.972  967  229         0.0  976  229
-1.930  929  153        -2.52  959  229          2.754  968  153         3.625  977  153
-9.969  930  153        -4.109  960  229         9.721  968  229
-7.94  930  229         -3.349  961  229         14.385  970  229
```

Trial 4

```
 0.717  977  229         0.588  985  153        -4.999  1000  229        13.35  1009  229
-1.38  978  153          0.116  985  229        -4.751  1001  153        4.998  1009  153
 0.379  978  229         0.158  986  153        -3.826  1001  229        2.647  1010  153
-1.748  979  153         0.626  993  153        -3.465  1003  153        4.951  1010  229
 0.348  979  229         0.136  992  229        -4.649  1003  229        8.709  1011  153
-1.141  980  153         1.844  994  153        -4.665  1004  153        2.629  1011  229
 0.0  980  229           0.799  993  229        -8.159  1004  229        12.901  1012  153
 0.939  981  229        -2.121  994  229        -11.3  1005  153         11.422  1012  229
-2.567  981  153         2.418  995  153         0.632  1005  229        15.214  1013  153
 0.747  982  153         0.359  995  229        -7.195  1006  153        7.539  1013  229
 0.266  982  229         1.835  996  153         1.984  1006  229        7.320  1014  153
 0.740  983  229        -1.802  996  229         13.107  1007  229       -1.786  1014  229
 0.550  983  153         0.0  999  153          3.889  1007  153         13.95  1015  153
 0.50  984  229         -5.962  999  229         11.670  1008  229       0.0  1015  229
 0.298  984  153        -1.455  1000  153        1.602  1008  153
```

Trial 5

```
 4.474  1016  153        0.103  1017  229        0.208  1019  153        0.723  1021  153
 2.613  1017  153       -1.3  1018  153          0.625  1020  153        0.203  1022  153
 0.105  1016  229        0.628  1018  229        0.671  1019  229        0.52  1020  229
```

```
0. 62  1021  229        -3. 302  1031  153      -8. 920  1038  153      0. 26  1043  229
-1. 241  1023  153      -6. 146  1030  229      -4. 409  1031  229      8. 596  1044  153
0. 143  1022  229       -4. 273  1032  153      1. 213  1039  153       5. 742  1044  229
0. 803  1023  229       -10. 628  1033  153     -1. 821  1032  229      13. 138  1045  153
0. 104  1025  229       -7. 613  1034  153      4. 462  1040  153       1. 653  1045  229
0. 651  1026  229       1. 127  1034  229       4. 41  1041  153        8. 381  1046  153
6. 132  1027  153       -6. 272  1035  153      -1. 626  1033  229      3. 904  1046  229
-1. 754  1027  229      1. 965  1035  229       2. 401  1042  153       20. 643  1047  153
2. 617  1028  153       -1. 921  1036  153      3. 601  1039  229       0. 0  1047  229
0. 899  1028  229       -11. 664  1036  229     14. 623  1040  229      6. 91  1048  153
-2. 574  1029  153      -11. 920  1037  153     8. 3  1041  229         0. 124  1048  229
-3. 492  1029  229      -4. 111  1037  229      1. 220  1043  153
0. 82  1030  153        8. 642  1038  229       3. 99  1042  229
```