

WIDAR – Wireless Intrusion Detection and Ranging

Daniel J. Gieseeman, PhD student, Electrical and Computer Engineering

Abstract—Sales of 802.11 and Bluetooth wireless local area network technologies are tremendous areas of activity in the consumer and commercial networking market. With any kind of wireless communications, attention to security becomes more important as physical access is not required for an adversary to mount attacks. Common wireless security measures mainly emphasize the integration of encryption and authentication layers to defend against and deter an attacker, with little focus given to discovering and physically locating a hidden malicious transmitter. This paper describes a system of hardware and software for the passive scanning, detection, and location of active attacks against a wireless local area network. Such a system is considered novel in that the detection components utilize a software positioned high-gain antenna to actively scan for and analyze radio signals. The document begins with an introduction to the topic of wireless network attack attribution and a review of relevant research. Next, the need for further research in the area of active scanning instrumentation is stated. In response to this need, the author presents the design and implementation of a proof-of-concept instrument. A theory of operation section is included detailing how a spatially distributed network of these devices would detect and triangulate an active attacker. This system is called WIDAR by the author, short for Wireless Intrusion Detection and Ranging. A series of operational tests are planned to evaluate the performance and effectiveness of the experimental device. The system test methodology is described in detail. The paper concludes with recommendations for system improvements and offers direction for future research.

Index Terms—attack, attribution, intrusion detection, networks, WIDAR, wireless

I. BACKGROUND AND OBJECTIVE OF THIS RESEARCH

A. Wireless Networks and Security

THE ubiquity of wireless local area network equipment has freed consumer and commercial grade communications technologies from the physical constraints imposed by wired connections. This freedom allows the home user to surf the web from the comfort of a sofa, or the business professional to send email while relaxing in the office courtyard.

Such a mobility-freeing technology comes at the cost of higher security risk than that of the traditional wired network.

This is due to the fact that wired infrastructure can be physically secured (e.g. placed in a locked wiring closet) whereas wireless network infrastructure, by its very nature, permits a malicious user to launch network-based attacks from anywhere within range of wireless transmission equipment. The operating range of wireless equipment typically includes locations outside the physical premises of an office building or home office. Hence, we see attack vectors targeted at wireless networks described in the literature, such as the ‘Parking Lot Attack’ and ‘WiFi Drive-By Attack’ [1].

Wireless vendors have responded to increased vulnerability by designing and integrating encryption and authentication components into wireless protocols. One of the most commonly deployed wireless local area network (WLAN) protocols, the 802.11 standard and its industry association moniker, wifi offers the infamous Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA) as examples of encryption and authentication protocols [2]. Such systems offer limited data link layer protections against a passive eavesdropper, but fall short against active denial of service (ADOS) attacks and active initial vector sniffing attacks designed to rapidly capture enough encrypted packets to crypt-analyze and deduce the wireless encryption key in use for a service domain [3].

Furthermore, a new class of attacks targeting poorly implemented wireless device driver implementations has also made headlines [4]. These attacks, when coupled with recently released wireless device driver stack fingerprinting software (capable of identifying driver version and revision numbers), allow attackers to identify a specific wireless card and then mix and match vulnerability exploits to a targeted device [5].

The potential for ADOS and targeted device driver attacks (against critical wireless infrastructure) points to the need for the capability to detect and locate an active attacker in real time. As such, having a transmitter location capability would serve to augment other types of forensic evidence collected during an attack, or even serve to tactically direct law enforcement operations. We must then look for solutions that solve the transmitter location problem.

December 03, 2006.

Daniel J. Gieseeman is a PhD student in Electrical and Computer Engineering at Iowa State University. e-mail: dgiesema@iastate.edu.

B. Review of Current Research

There are several examples of such wireless device location systems emerging in the marketplace. Cisco's Wireless Location Appliance advertises the capability to track the location of thousands of wireless 802.11 devices within a service set domain [6]. This system relies on the creation of a database of the signal propagation characteristics of potentially thousands of sample points within a service set domain.

Additionally, a white-list of allowed devices is required to be maintained, so that unauthorized devices can be flagged and located based on a signal fingerprint lookup found in the signal propagation database. This system has the limitation of the increased maintenance overhead imposed by maintaining the white-list, and the one-time cost of mapping the signal attenuation characteristics of a facility.

Berkeley Varitronics Systems offers the Yellow Jacket, a commercial signal analyzer allowing a skilled analyst using the system, along with a directional antenna, to track down the location of a hidden wireless station [7]. The drawback to using a system such as this is the skill level necessary for interpreting signals and the high cost of deploying such a system.

Even more interesting is a document published by Interlink Technologies detailing the use of signal attenuation heuristics to locate a transmitting wireless NIC or access points [8]. The whitepaper describes a system using only the signal loss characteristics detected by a mobile passive listener to model the physical location of a WLAN transmitter. The approach is novel in that directional antennas are not used; instead a signal attenuation model is utilized to guess the distance a transmitter is located from a receiver.

Combining signal readings from several locations about an area, and assuming that the signal power of the transmitter is known and decays in a well understood manner, the author then describes equations to determine the location of a hidden transmitter. The conclusions of the whitepaper show a high degree of success in estimating the location of a transmitter with a known control position.

Drawbacks to this approach are the requirement for an accurate model of signal attenuation for the specific operational environment, and the need to know the effective broadcast power of the attacking transmitter (something difficult to know ahead of time, in a real world scenario). The author recommends the need for enhanced signal propagation modeling of the facility being monitored, similar to the Cisco product. The author also recommends the use of directional antennas for "real triangulation" of a transmitter.

Further study of the use of directional antennas led to a paper published by researchers at the University of New

Orleans. In this paper the authors successfully demonstrate the use of a high-gain antenna system for active transmitter localization [9]. The authors combine an anomaly and signature-based Wireless Intrusion Detection System (WIDS) with the several types of high-gain antennas to locate a wireless intruder. Also of interest is the graphical analysis method used by researchers to plot signal strength levels and antenna angular bearing on a polar coordinate system.

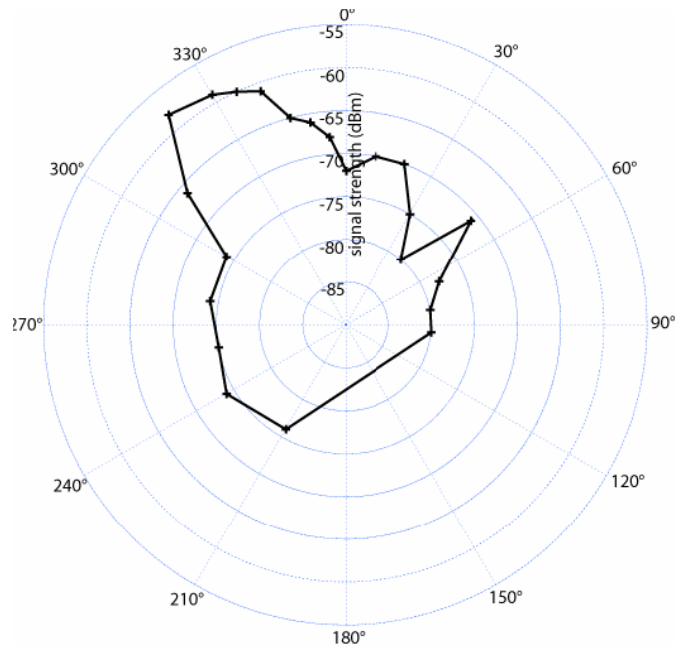


Fig. 1. Graphical Signal Strength Plot. This plot from the University of New Orleans study is graphically displaying data collected by pointing a directional antenna at a transmitter with bearing 330° from center about 150 feet away.

The University of New Orleans document concludes by repeating the utility of directional dish antennas as a means of accurately measuring the angular bearing of a receiver to a transmitter. The authors also mention their intended next step is motorizing the directional antenna for automated tracking of a transmitter. A drawback of this approach is that only active transmissions are detected. A passive eavesdropper cannot be located with directional-antenna based location methods.

C. Statement of Research Objective

It is the intent of this research to build upon the directional-antenna based location methodologies employed with success in the previously discussed overview of relevant research. Towards this end, the remainder of this document describes the design, implementation, and evaluation plan for a motorized device capable of active malicious-transmitter scanning using low-cost, consumer-grade equipment and freely available or custom-developed software. The hardware and software system described by this effort is called WIDAR by the author, short for Wireless Intrusion Detection and Ranging. The complete design for the WIDAR transmitter location system is documented in this paper, although, to demonstrate

proof-of-concept, only a single prototype WIDAR sensor was constructed for this research.

II. SYSTEM ARCHITECTURE

A. System Concepts

The WIDAR system for wireless intruder location is composed of a network of two or more WIDAR sensor devices interlinked by a distribution system (in our case, a wired local area network) to centralized command and control software. Conceptually, spatially-distributed WIDAR sensors (with known locations) work in tandem to track separate angular vectors of strongest signal strength for an intruding transmitter (with an unknown location). Signal strength vector information collected by each sensor is communicated over the distribution system for analysis by the command and control sub-system in real time.

The command and control sub-system performs trigonometric analysis on the signal strength vectors to triangulate a candidate location for a detected adversary. Analysis results are accessible to client applications via an application programming interface (API) designed for the WIDAR system (e.g. the WIDAR API). All of these operations are described in greater technical detail in the Theory of Operation section of this paper.

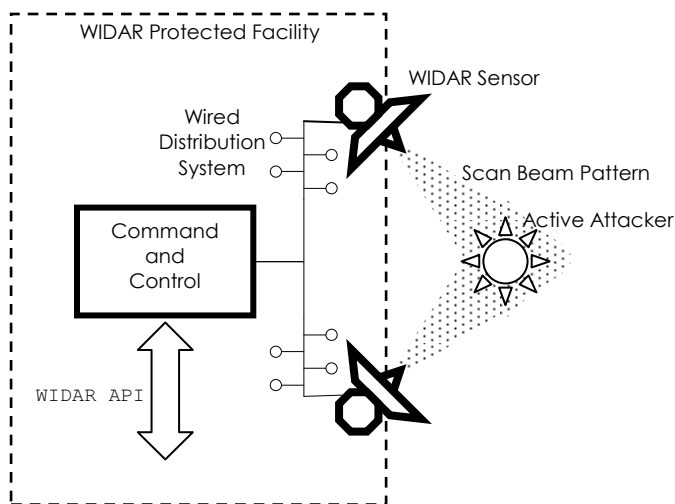


Fig. 2. WIDAR Conceptual Architecture. Shown are two WIDAR sensors communicating using the Distribution System with the Command and Control system to triangulate the location of an Active Attacker. The WIDAR API is provided for clients to integrate and use the WIDAR system (e.g. as the transmitter location component of NIDS software such as AirSnort).

B. The WIDAR Sensor

A WIDAR sensor is composed of a motorized chassis mounted on a fixed base and contained within a weatherproof and camouflaged housing. The intent of concealment is to hide the plain sight view of forensic signals analysis hardware in operation at a facility from an alert and prepared adversary. For the prototype developed during this research, the WIDAR

sensor chassis is capable of actively positioning a 17dBi 2.4 GHz directional dish antenna using two high-torque DC gear motors under software control.

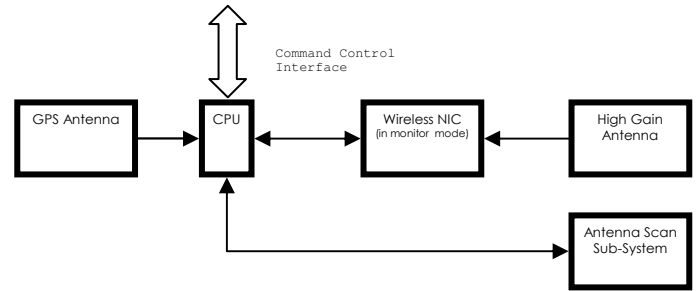


Fig. 3. Block Diagram for a single WIDAR Sensor. The WIDAR sensor communicates with the Command and Control System through the Command Control Interface. Sensors learn their x and y position using GPS. The WIDAR sensor CPU positions the sensor dish antenna using the Antenna Scan Sub-System.

The dish antenna feed horn connects to a network interface card (NIC) running in passive monitor mode on a CPU hosting a Linux operating system. This host is focused on the collection of signal strength and angular state information for detected wireless stations deemed malicious by the command and control sub-system. The system design also specifies that a GPS receiver can also be connected to the CPU to provide the WIDAR sensor a projected x, y coordinate pair for position.

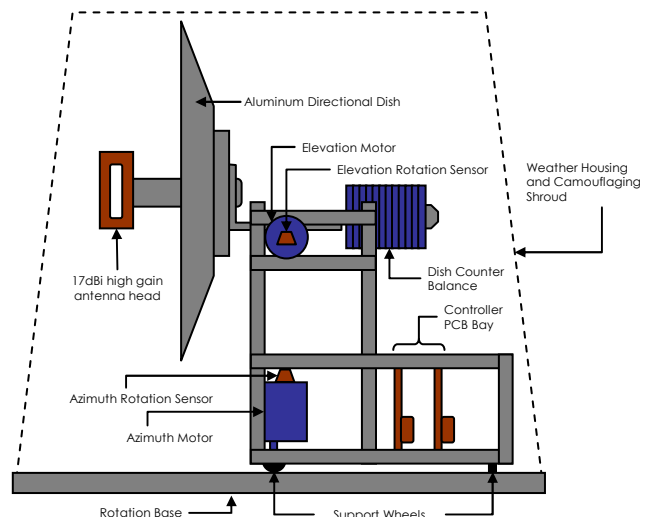


Fig. 4. Physical Model of a WIDAR Sensor. Shown are the primary components composing the WIDAR sensor.

The WIDAR sensor is built onto a framework of hobby-grade ABS plastic. Three ball-bearing support wheels allow the chassis to rotate easily about a central axis on a fixed base. The chassis mounts two motors for the control of the directional dish azimuth and elevation. Due to the weight of the antenna dish, the elevation rotation axis is balanced by an

attached counterweight.

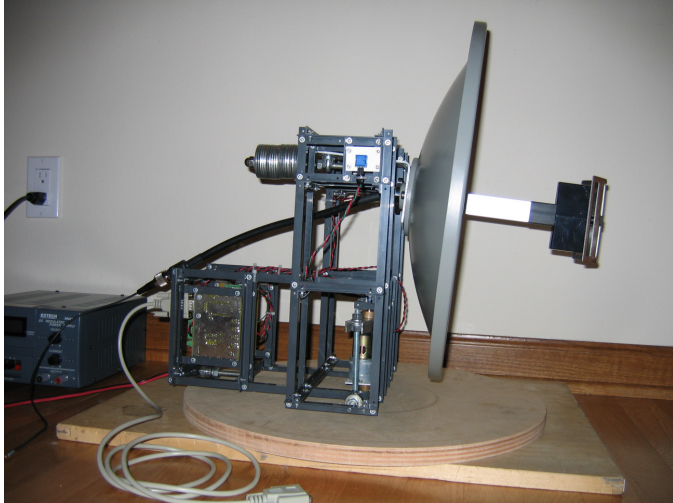


Fig. 5. Prototype WIDAR Sensor. The main dish and feed horn are shown, along with the ABS plastic frame and blue rotation sensing potentiometer.

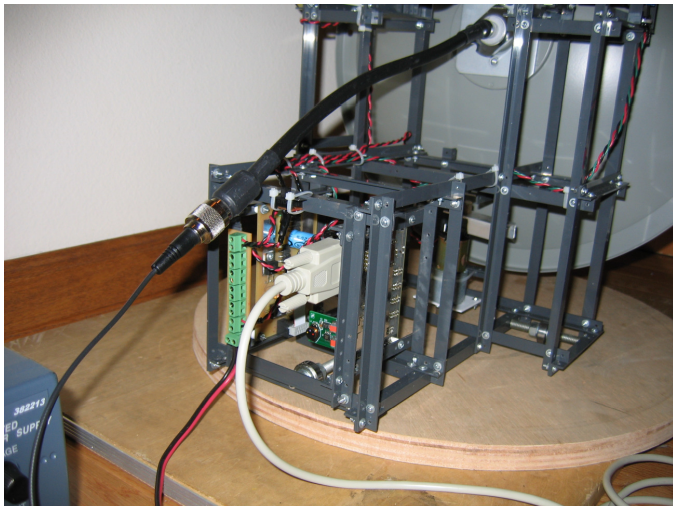


Fig. 6. Rear close-up of Prototype WIDAR Sensor. The PCB controller bay is visible in the foreground, along with the serial interface cable, antenna 50 ohm connector cable, and rear ball-bearing support wheel.

C. Antenna Scan Sub-System

Antenna pan and tilt operations are the responsibility of the Antenna Scan Sub-System. Dual elevation and azimuth motors are controlled by an 8-bit microcontroller and h-bridge combination housed in the PCB bay of the WIDAR sensor. Speed control is accomplished via pulse width modulation (PWM) of the enable pin of the respective h-bridge driver channel.

Direct responsibility for PWM control of the motors is offloaded from the microcontroller to a dedicated PIC-based pulse co-processor, freeing the microcontroller to perform higher level management functions such as communications with the command and control system and the collection of antenna azimuth/elevation data. Both the elevation and azimuth axes are coupled to rotation-sensing potentiometers read by a single onboard two-channel, 12-bit analog digital

converter (ADC). The ADC provides the onboard microcontroller accurate angular rotation parametric data which are in turn made available for polling by the command and control software via an RS-232c serial data link.

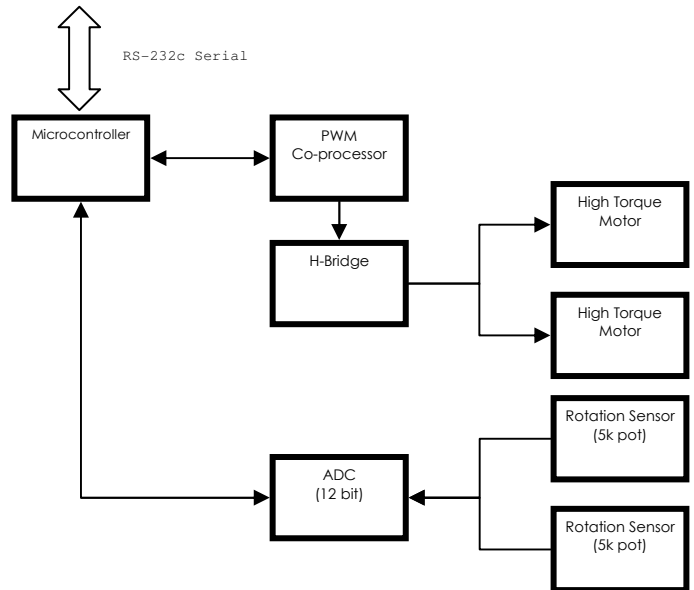


Fig. 7. Block Diagram of the WIDAR Sensor Antenna Scan Sub-System. The scan sub-system is responsible for the low-level control of motor speed and direction, along with sensing the antenna dish azimuth and elevation.

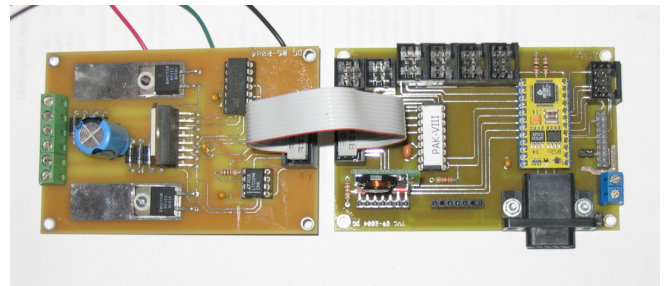


Fig. 8. Photo of the microcontroller (right) and h-bridge (left) printed circuit boards. The boards were designed using the software of an online board manufacturer. The microcontroller is the yellow component located in the center right. In the lower right of photo is the DB-9 serial connector. The h-bridge receives command signals via a ribbon-cable connection to the onboard computer board.

D. WIDAR Sensor Software

Software modules comprise the intelligent components of the WIDAR system. As an overview, low-level microcontroller firmware (called the Scan Controller firmware) deployed on each WIDAR sensor move the antenna directional dish through a pre-programmed scan pattern to log and timestamp dish attitude (azimuth and elevation). The scan pattern is determined PC implemented software (called the Antenna Scan Driver) which communicates directly with the Scan Controller firmware to position the WIDAR sensor antenna.

Concurrently, the WIDAR sensor host CPU runs network packet capture software to collect and timestamp signal

strength information for stations actively transmitting in range of the WIDAR sensor device. The host CPU for sensor transmits collected dish attitude and signal strength information to system command and control software using the distribution system backbone. Since both datasets contain timestamp information from the same host CPU clock, the command and control software can temporally join these data to provide integrated attitude and signal strength information to analysis software via the WIDAR API. For prototyping, an analysis tool was developed for graphically viewing collected signal strength data (called the WIDAR Signal Analyzer).

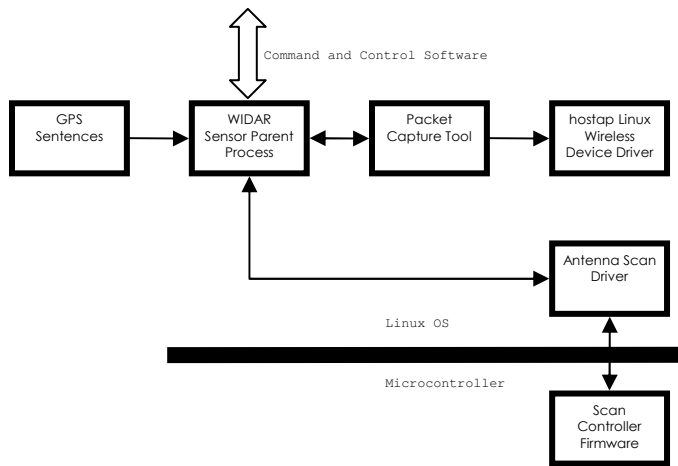


Fig. 9. Block Diagram of WIDAR Sensor Software. Most of the software is implemented on a Linux operating system due to the capabilities of this operating system to place a wireless NIC in passive monitor mode, and the ability to access layer 2 radio packet headers containing per packet signal strength information received from the wireless card device driver.

1) Scan Controller Firmware

Low-level hardware operations responsible for motor speed and direction control and rotation sensing are under the control of the Scan Controller firmware. The firmware implemented logic acts as a server to the master Antenna Scan Driver software client. Firmware logic opens an RS-232 serial communications port of the microcontroller and listens for commands.

For the test prototype, microcontroller firmware received data arriving at 9600 baud, 8 bit characters, with no parity, and one stop bit. The Scan Controller firmware was implemented in Parallax Basic, the development language used by the Basic Stamp 2p (BS2p) microcontroller. The BS2p microcontroller features a built-in RS-232 line driver transceiver handling the voltage shifts from RS-232 to the TTL/CMOS levels used by the 8 bit controller. Appendix A contains a listing of the Scan Controller firmware source code. Appendix B contains a diagram of firmware control decision flow.

The Scan Controller firmware implements an open feedback loop driving goal-oriented position and speed seeking logic. In this configuration, target position data

received from a master unit (in our case the Linux host CPU via RS-232) are required to drive the open loop. If target azimuth and elevation position data are not received at least once per second, a loop watchdog timer causes motor operation to stop, then returns the loop to the listening state waiting for more target position data. In prototype development testing, the open-loop was seen to iterate many times per second, and the watchdog timer was only triggered when the serial communications was removed.

As mentioned, a single loop iteration is initiated by a master host sending the microcontroller a request to position the WIDAR sensor motors. A simple three-byte command protocol is used to receive motor position requests. The first byte is a header byte, and must be an 'F' for the microcontroller to recognize and act upon the received data. The second and third bytes are the requested azimuth and elevation positions.

At the start of each of loop iteration, position sensing is first performed so logic can determine if a motor rotation axis is reaching the target position. 12-bit ADC values are scaled to the 8-bit position requests by dividing by 16. This procedure has the added benefit of calming sensitive shaft position readings, minimizing processor calls for un-necessary ultra-fine gross position adjustments.

The position sensing logic will place a motor position into one of four categories:

- Motor is at the correct position; stop the motor if it is active.
- Motor is not at the correct position, but is traveling the right direction; do nothing.
- Motor is not at the correct position, but is traveling the wrong direction; slow down the motor to zero speed, before switching motor direction.
- Motor position is outside a logically legal range; stop the motor immediately.

The loop concludes by repeating the process for the remaining axis motor, and then returning current axis rotation information data to the connected master unit, via the serial communications port.

The open-loop design of this implementation allows for on-the-fly position target data changes, such as the emergency stopping of an axis motor. By contrast, in a closed-loop configuration, the firmware would receive a target position and not yield control to a master CPU until the position was obtained. The open-loop scenario implemented in the WIDAR firmware design provides near-instantaneous response to changes in antenna target position requested by a master host.

Adaptive motor direction and speed control routines are

also components of the microcontroller firmware implementation. The goal-seeking design of the open-loop does not instantaneously switch motor direction while a motor is in motion, or change motor speed from all-stop suddenly to full-speed. Instead, ramping logic incrementally steps motor speed to a globally defined maximum position attainment speed, by adding a globally defined speed change increment.

The open-loop must be iterated several times before maximum speed is obtained, the exact number being determined by the max speed and speed increment constants. At any time during operation of the open-loop, the speed target may be altered, to reflect a change in position target, or the execution of an emergency stop by the system.

To eliminate sudden, jerking motor stops (except for emergency stops), the firmware logic will adjust a motor speed from the max speed constant to a globally defined fine-adjust speed, when a motor axis position is determined to be in proximity to the axis target position. Proximity sensitivity is set using a globally defined motor proximity constant. In the prototype, the fine adjust speed was determined (through hardware experimentation) to be the minimum speed value necessary for a viewer to see active antenna movement.

2) Antenna Scan Driver

The Antenna Scan Driver is a client of the Scan Controller firmware and implements a complete scanning tool in the C language on the Fedora Core 5, Linux Operating System. This module implements antenna-scan patterns for driving the open-loop of the WIDAR firmware. In addition, the driver manages serial communications with the slave microcontroller of the WIDAR sensor hardware, and most importantly logs and timestamps antenna position information (current azimuth and elevation rotation) output at each loop iteration. The Driver utilized POSIX compatible serial port routines to interface with the WIDAR antenna position hardware using the RS-232 port of the Linux host machine. Appendix D contains a listing of the Antenna Scan Driver source code.

3) Packet Capture Tool and Network Interface Card

The Ethereal Network Analyzer currently serves as the network packet capture tool for the WIDAR system. In the future, it is desirable to integrate the Antenna Scan Driver and packet capture engine into a single driver utility. The Linux operating system provides the hostap driver for use with the Senao NL-2511CD PLUS EXT2 wireless network interface card.



Fig. 10. Senao NL-2511CD PLUS EXT2 wireless network interface card. The antenna pigtail and connector are also shown.

This card utilizes the Intersil Prism2 chipset and, as such, the hostap device driver is able to provide the Prism Monitoring Header to the libpcap library linked by Ethereal for packet capture. The Prism header is an important prerequisite to the prototype WIDAR system, as additional Layer 2 information in this header packet provides access to received signal strength information on a per packet basis.

For this research, Ethereal runs (containing captured packet signal strength data) were saved in the bpf pcap format for analysis by higher-level tools. Captured traffic was also filtered to contain only 802.11 management and data frames, to simplify the parsing logic needed to process the capture data.

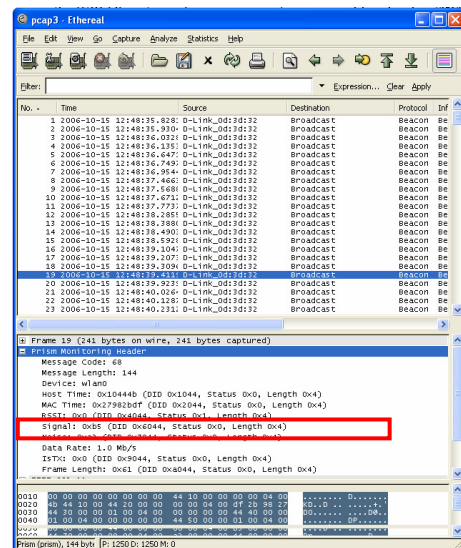


Fig. 11. Ethereal Screen Capture. The PRISM Monitor Mode Header is shown and the Signal Strength field is highlighted in red. This field and accompanying timestamp and source MAC information are critical components of the WIDAR transmitter location methodology.

4) WIDAR Signal Analyzer Software

The WIDAR Signal Analyzer software tool was developed to test the single WIDAR sensor developed for this research. The signal analysis software provides a graphical display of the integrated antenna position and signal strength information, with the capability of showing the compass bearing of the antenna chassis, the current sweep position of dish (azimuth

and elevation, and a trace of recent signal strength data.

The WIDAR Signal Analyzer is implemented in Microsoft Visual Basic 6. The software is driven by log files created by both the Antenna Scan Driver and Packet Capture Tool. A data import component processes data from these capture files for storage in a Microsoft Access relational database format. The WIDAR Signal Analyzer tool can dynamically query and display vectorized signal data for any MAC address of interest found in the scan capture period.

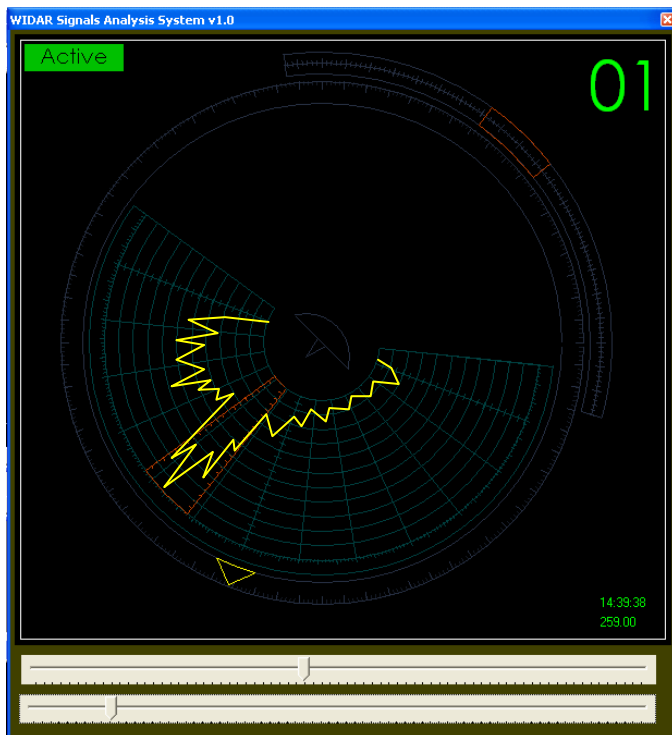


Fig. 12. WIDAR Signals Analysis Client GUI. Shown is a prototype GUI developed to display signal strength and bearing information collected by a WIDAR Sensor.

III. SYSTEM THEORY OF OPERATION

This section specifies how the previously described system components interoperate to passively identify and triangulate an actively transmitting attacker.

A. Prerequisites

It is intended that the WIDAR system would protect a facility (hereafter called the WIDAR protected facility) by operating as the location processing component of a wireless intrusion detection system (WIDS) and possibly as the passive packet gathering tool for the WIDS. In the first mode (active scanning), the system is actively seeking to determine the position of an adversary. In the second mode (generalized scanning) the system is simply panning the network of WIDAR sensor antennas through their full field-of-view range. This process supplies the WIDS with additional gathered packets from a high-gain source. The WIDS may also need to gather packets from non-directional sources, to support more robust stateful inspection logic.

The setup, configuration, and underlying logic implementing the WIDS and detecting a wireless intrusion are beyond the scope of this paper. For this research it is assumed that a suspicious or malicious transmitter has been identified and subsequently, the WIDAR system command and control software is provided a list of target MAC addresses for which to attempt transmitter location.

B. System Deployment and Activation

In response to an active location or generalized scanning request, the WIDAR command and control system initiates active scanning, sending control signal packets via the distribution system to a network of two or more WIDAR sensors. The sensors are strategically located about the WIDAR protected facility perimeter. It is intended that WIDAR Sensors are deployed in a manner such that antenna scan patterns spatially overlap to support the trigonometric analysis method used for location estimation.

Each sensor passively monitors target packets transmitted by a target MAC address, recording signal strength, timestamp, and antenna position information for relay via the distribution system to command and control software. These functions are specified for implementation as a single process on the WIDAR sensor host CPU.

C. Modes of Operation

As mentioned, the WIDAR sensor is pre-programmed with a variety of search and scan modes. Modes range in function from simple signals gathering scanning movements covering the full range of sensor field-of-view, to an active homing mode with the objective of intelligently seeking the bearing of maximized signal strength for a target MAC address. Scanning algorithm logic is implemented within the Antenna Scan Driver component of the WIDAR system. The particular mode of operation employed by the WIDAR sensor is selected under the direction of master command and control software.

The primary mode of operation is a simple field-of-view scan, causing the antenna dish to be moved through the full range of azimuth, followed by an incremental change in elevation, and again followed by a full range of azimuth. This pattern is repeated through the full elevation range, and then reversed to return back to the loop beginning, where the scan is then repeated.

The active signal strength maximizing mode of operation employs a search algorithm incorporating logic capable of making intelligent determinations about signal strength changes as antenna orientation is modified. Logic is given the objective of finding the angular bearing which maximizes the received strength of a Target MAC address.

D. The Distribution System

While not implemented as part of the system prototype, the WIDAR system design integrates a distribution system enabling distributed WIDAR sensors to communicate over a network backbone to centralized command and control software. It is intended that the distribution system be implemented using wired 802.3 CSMA/CD (Ethernet) local area networking hardware. The use of wired LAN technology is specified in lieu of wireless communications devices, to prevent the possibility that the distribution system itself is attacked wirelessly, potentially disabling or denying service, rendering the protective capabilities offered by the WIDAR system useless.

The WIDAR sensor parent process streams collected signals data to the central command and control application using a custom UDP implemented protocol. The Command and Control application should host a UDP listener, open on UDP port 33,333 of the Command and Control host network stack, and capable of logging packets to a centralized repository for analysis.

E. The WIDAR Triangulation Method

Trigonometric analysis is used to triangulate a target transmitter. This logic is based on the mathematics of the triangle, and employs the laws of sines and cosines, in addition to equations derived from these laws for the conversion of Cartesian coordinate pairs (x, y) to polar coordinates (r, θ).

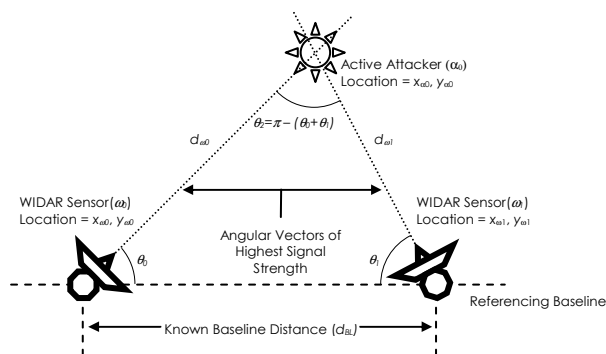


Fig. 13. Schematic Diagram of the WIDAR Triangulation Method. Shown are the basic trigonometric attributes required for triangulation of an active transmitter. The distributed WIDAR sensors gather the two required angles, θ_0 and θ_1 , used to solve the location equations.

The base input requirements for the triangulation logic are the maximum signal strength bearings collected from two WIDAR sensors (ω_0 and ω_1) having overlapping coverage field-of-views. The third angle necessary for triangulation (ω_2) can be calculated by subtracting the sum of the two WIDAR sensor collected angles from π radians (in Euclidean geometry the sum of the angles of a triangle must be π radians).

These two bearings (called θ_0 and θ_1) are represented as angular rotations about a common baseline segment (BL). It is

assumed as given that each WIDAR sensor has a known location designated by x and y coordinates ($x_{\omega0}$, $y_{\omega0}$, $x_{\omega1}$, $y_{\omega1}$ respectively). The baseline distance (d_{BL}) between the two WIDAR sensors can be calculated using the Cartesian distance formula (equation 1).

Given two vector angles pointing to an actively transmitting attacker, and a known distance between the two angles, the law of sines (equation 2) can be used to calculate the distance to the attacker ($d_{\omega0}$). This distance can, in turn, be used to calculate the x and y coordinates of the attacker using polar coordinate to Cartesian coordinate formulas derived from the laws of sines and cosines (equations 3 and 4).

$$d_{BL} = \sqrt{(y_{\omega1} - y_{\omega0})^2 + (x_{\omega1} - x_{\omega0})^2} \quad (1)$$

$$d_{\omega0} = \frac{\sin(\theta_1) * d_{BL}}{\sin(\theta_2)} \quad (2)$$

$$x_{\omega0} = d_{\omega0} * \cos(\theta_0) \quad (3)$$

$$y_{\omega0} = d_{\omega0} * \sin(\theta_0) \quad (4)$$

F. The WIDAR User Interface

The WIDAR system should provide software tools offering a user interface to graphically depict the estimated location of a Target MAC address, along with any relevant signal strength and bearing information necessary for signal analysis. This user interface should also permit the manual control of the WIDAR sensor array for real-time monitoring of a location by a trained analyst. For this research, the WIDAR Signal Analyzer serves this purpose.

The WIDAR system design also specifies that each WIDAR sensor should integrate with a video capture system to augment collected signals data with supplemental video evidence of a potential crime, and to provide the intrusion system with generalized video surveillance functions. It is desirable that software video capture display components be directly integrated with the signals analysis WIDAR user interface, although this feature was not implemented or tested for the current WIDAR prototype.

IV. SYSTEM TEST METHODOLOGY

This section describes a series of three test scenarios designed to test the prototype WIDAR device. The setup conditions for each scenario are presented, and descriptions of specific test objectives are detailed. The section concludes by explaining the planned schedule of testing and analysis of results. In the scenarios, location determination should be performed with a measuring wheel for indoor testing, and using GPS for outdoor testing.

A. Access Point Triangulation Test

The first test of the WIDAR device intends to test the ability of the single prototype sensor to find the location of a known 802.11 access point. This scenario has two sub-components: one performed inside an office building or home, and the other run outdoors with clear lines of sight to the access point.

In this test scenario, the MAC address of the transmitter is known, and it is intended that the sensor device will be deployed for the collection of two sets of data along a known baseline distance. The first set of data will simulate the results of WIDAR sensor 0, and the second set will simulate the results of WIDAR sensor 1. The resulting data files will be imported into the WIDAR Signal Analyzer for graphic analysis and evaluation of effectiveness for a known transmitter location.

B. Parking Lot Attack Transmitter Location Test

The second WIDAR device test aims to evaluate the ability of the WIDAR system to identify an attacker with unknown location. In this scenario, the MAC address of the transmitter is known, but the location of the adversary is not. It is intended that testing for this scenario be performed in the parking lot of an office building, with the WIDAR device deployed on opposite ends of the building for the collection of two data sets simulating two WIDAR devices, as in Scenario A.

As in previous tests, the resulting signal strength and antenna position data files will be imported into the WIDAR Signal Analyzer for location analysis. The resulting location determination will be compared with the known location of the attacking host, which should not be revealed to the WIDAR signals analyst until a location guess has been made.

C. Spoofed Duplicate MAC Address Test

The objective of the final test scenario is to evaluate the behavior of the WIDAR sensor signals gathering logic when two transmitters are active, each using the same MAC address. For this scenario, a control host and an attacking host are necessary. The intention is to simulate a replay attack, which commonly uses the replay of a captured encrypted data frame, to actively cause an Access Point to produce many encrypted response packets, with the goal of ultimately obtaining enough captured packets for an attacker to statistically deduce the WEP encryption key being used for a service set. To accomplish this, the attacking host will be configured to listen for and capture an encrypted ARP frame generated by the control host (it will be 54 bytes long). The attacking host will then replay this frame, while the control host is configured to generate normal traffic, such as a repetitive ping request.

This scenario should include sub-component tests; one test with two transmitters being closely aligned along a common bearing (for one of the test WIDAR sensors), and with two transmitters being widely dispersed. bAs in previous tests, the resulting signal strength and antenna position data files will be imported into the WIDAR Signal Analyzer for location analysis. The resulting location determination will be compared with the known location of the attacking host and the control host.

V. ANALYSIS OF RESULTS

Test results stemming from the implementation of the testing methodology described previously will be presented to the Computer Engineering 536 Forensics Group on December 14th, 2006.

VI. CONCLUSIONS AND RECOMMENDATIONS

This paper documents research performed as term paper work for the Computer Engineering 536 Computer and Network Forensics course Fall 2006 offering at Iowa State University. The bulk of the many, many hours spent during this effort were focused on getting an operational prototype WIDAR sensor functional. The system design discussed in this paper serves to describe the intent of the Wireless Intrusion Detection and Ranging system, and specify a theory of operation for the system.

Future time spent researching this system will focus on testing the now operational WIDAR sensor, and refining the system to be more tightly integrated. Apart from testing, examples of intended future effort include:

- Modify the WIDAR Antenna Scan Driver to include logic accessing the appropriate device driver ioctl function to support channel hopping behaviors.
- Integrate the collection of signal strength and antenna attitude information into a single process utilizing libpcap for packet capture.
- Implement a message switch capable of relaying command and control instructions to WIDAR sensors, and relay of signal capture information to the WIDAR Signal Analyzer in real time.
- Explore the modification of the hostap wireless card driver for enhanced hardware level access to the Intersil chipset.

Recommendations for future research include the usage of the system architecture for analysis of Bluetooth and other wireless protocols, along with the use of the WIDAR system with new methods for wireless 802.11 device fingerprinting described in [5].

The author wishes to thank Dr. Yong Guan, the course

instructor, for the opportunity to explore this topic as part of the Computer and Network forensics course term paper. Much knowledge was learned following the course of researching, designing, fabricating, and documenting the WIDAR sensor prototype, including enhanced Linux kernel and device driver understanding, low level microcontroller programming, analog to digital logic, in addition to the GUI programming and network forensics experience obtained.

REFERENCES

- [1] S.S. Miller, *Wifi Security*," McGraw-Hill Networking, 2003.
- [2] M. Gast, *802.11 Wireless Networks: the definitive guide*. O'Reilly, 2005.
- [3] M. Ossmann, *WEP: Dead Again (Parts I and II)*. Available: <http://www.securityfocus.com/infocus/1814>
- [4] D. Maynor, *Beginner's Guide to Wireless Auditing*," Available: <http://www.securityfocus.com/infocus/1877>
- [5] J. P. Elch, *Fingerprinting 802.11 Devices*, 2006 Master's Thesis, Naval Postgraduate School.
- [6] Cisco Wireless Location System product brochure. Available: http://www.cisco.com/application/pdf/en/us/guest/products/ps6386/c1031/cdccont_0900aecd80473268.pdf
- [7] YellowJacket 802.11b/g handheld spectrum analyzer. Available: <http://www.bvsystems.com/Products/WLAN/YJ802.11bg/YJ802.11bg.htm>
- [8] Interlink Technologies Whitepaper. *A Practical Approach to Identifying and Tracking Unauthorized 802.11 Cards and Access Points* Available: http://www.interlinknetworks.com/graphics/news/wireless_detection_and_tracking.pdf
- [9] F. Adelstein, P. Alla, *Physically Locating Wireless Intruders*, Available: <http://www.cs.uno.edu/~golden/Stuff/WIDSpaperIEEE.pdf>

APPENDIX A - ANTENNA SCAN CONTROLLER FIRMWARE SOURCE

```

' {$STAMP BS2p}
' {$PBASIC 2.5}
' {$PORT COM5}

CTL_CMD_DAT  VAR Byte(3)           'Command Packet for TX/RX
CTL_CMD_TMO  CON 1000             'Command Packet Receive Timeout Value (We want this to be 1 second)

ADC_PIN_CS   CON 13               'ADC Chip Select; 0 = active
ADC_PIN_CLK  CON 14               'ADC Clock Pin; out on rising, in on falling edge.
ADC_PIN_DAT  CON 15               'ADC Data I/O PIN
ADC_CFG_DAT  VAR Nib              'ADC Config Bits

AZI_PIN_DIR  CON 4                'Azimuth PAK Direction Channel
AZI_PIN_SPD  CON 7                'Azimuth PAK Speed Channel
AZI_POS_TGT  VAR Byte              'Azimuth Motor Position Target
AZI_POS_MIN  CON 0                'Azimuth Lower Sense Position Logical Limit
AZI_POS_MAX  CON 255              'Azimuth Upper Sense Position Logical Limit
AZI_SPD_CUR  VAR Byte              'Azimuth Speed Current
AZI_SPD_LST  VAR Byte              'Azimuth Speed Last Sent to PAK Pulse Co-Processor
AZI_SPD_TGT  VAR Byte              'Azimuth Speed Target
AZI_DIR_CUR  VAR Bit              'Azimuth Direction Current

ELE_PIN_DIR  CON 5                'Elevation PAK Direction Channel
ELE_PIN_SPD  CON 6                'Elevation PAK Speed Channel
ELE_POS_TGT  VAR Byte              'Elevation Motor Position Target
ELE_POS_MIN  CON 85               'Elevation Lower Sense Position Logical Limit
ELE_POS_MAX  CON 135              'Elevation Upper Sense Position Logical Limit
ELE_SPD_CUR  VAR Byte              'Elevation Speed Current
ELE_SPD_LST  VAR Byte              'Elevation Speed Last Sent to PAK Pulse Co-Processor
ELE_SPD_TGT  VAR Byte              'Elevation Speed Target
ELE_DIR_CUR  VAR Bit              'Elevation Direction Current

MTR_NUM_CUR  VAR ADC_CFG_DAT.BIT2 'Current Motor Number (0 = Azimuth, 1 = Elevation)
MTR_PIN_DIR  VAR Nib              'Current Motor PAK Direction Channel
MTR_PIN_SPD  VAR Nib              'Current Motor PAK Speed Channel
MTR_POS_CUR  VAR Byte              'Current Motor Position Current
MTR_POS_TGT  VAR Byte              'Current Motor Position Target
MTR_POS_MIN  VAR Byte              'Current Motor Lower Sense Position Logical Limit
MTR_POS_MAX  VAR Byte              'Current Motor Upper Sense Position Logical Limit
MTR_SPD_CUR  VAR Byte              'Current Motor Speed Current
MTR_SPD_LST  VAR Byte              'Current Motor Speed Last Sent to Pulse Co-Processor
MTR_SPD_TGT  VAR Byte              'Current Motor Speed Target
MTR_SPD_MAX  CON 34               'Current Motor Maximum Speed
MTR_SPD_ADJ  CON 33               'Current Motor Fine Adjust Speed
MTR_SPD_INC  CON 10               'Current Motor Speed Adjust Increment
MTR_SPD_STP  VAR Bit              'Current Motor All Stop Flag (0 = Normal Operation; 1 = Stop Motor Now)
MTR_DIR_CUR  VAR Bit              'Current Motor Direction Current (0 = Left, Down; 1= Right, Up)
MTR_PRX_FCT  CON 15               'Current Motor Position Sense Proximity Factor

PAK_PIN_DAT  CON 12               'PAK VIII Data I/O
PAK_PIN_CLK  CON 11               'PAK VIII Clock Pin
PAK_VAR_CHN  VAR Nib              'PAK VIII PAK_BLD_CMD CHANNEL
PAK_VAR_REG  VAR Nib              'PAK VIII PAK_BLD_CMD REGISTER

TMP_BYTE VAR Byte                'Scratch-Pad Byte Variable
TMP_WORD VAR Word                 'Scratch-Pad Word Variable
TMP_BIT VAR Bit                   'Scratch-Pad Bit Variable

'*****
MAIN:
'*****
  GOSUB STARTUP                    'Run Init Routines

  'Fall into Command Wait Routines

  PAUSE 100

'*****
CMD_WAIT:
'*****

  SERIN 16, 16884, CTL_CMD_TMO, CMD_TIMEOUT , [WAIT("F"), STR CTL_CMD_DAT\3] 'Listen for Command

  SEROUT 16,16884, ["STR", CR]

  FOR MTR_NUM_CUR = 0 TO 1          'Run through the motors and perform the position tracking logic.

    GOSUB READ_POSITION_SENSOR      'Update the position of the current motor (by setting MTR_POS_CUR)
    GOSUB SET_VALID_POS              'Either update the Target Position with a new Position, or use the old one, if the
new is invalid
    GOSUB SET_CUR_MTR_STATE          'Set the Current Motor Parameters to the state vars of the respective motor

    IF ABS (MTR_POS_TGT - MTR_POS_CUR) <=1 THEN 'If the Motor is at the target position then stop

      MTR_SPD_TGT = 1                'Set the Motor Speed Target to 1
      MTR_SPD_STP = 1                'Set the Motor Stop Bit to tell the speed routine to stop motor in this iteration

    ELSE

      GOSUB CHK_DIR                  'If the Motor is not at target position, then find out if the motor direction is
setup to go that way

      DIRSET:                          'Return Entry Point, if the Direction Bit is able to be Changed, we want to still
make a speed inc too
      IF TMP_BIT = 1 THEN              'The Motor is going the right direction

        GOSUB SET_TGT_SPD            'Set the Target Speed for the Motor

      ELSE

        IF MTR_SPD_CUR = 1 THEN        'The Motor is going the opposite direction, but not moving

```

```

    GOSUB FLIP_MTR_DIR           'Flip the motor direction
    TMP_BIT = 1                 'Change the CHK_DIR Bit to True (1)
    GOTO DIRSET                 'Return to the Decision Re-entry point

ELSE

    MTR_SPD_TGT = 1            'Set the Target Speed to 0

ENDIF

ENDIF

ENDIF

GOSUB UPDATE_SPD              'Increment the Speed toward the Speed Target

IF MTR_SPD_CUR <> MTR_SPD_LST THEN 'Only Send Speed Values to Co-Processor that are different than last request

    GOSUB SET_MTR_SPD         'Send the Speed Update
    GOSUB SYNCH_VARS          'Record any necessary changes into the correct State Variables

ENDIF

GOSUB SYNCH_POS               'Update the Appropriate Command Byte with the current sense position

NEXT

GOSUB SEND                    'Send Any Response Message

GOTO CMD_WAIT

'*****
CMD_TIMEOUT:
'*****

    GOSUB SEND

GOTO CMD_WAIT

'*****
UPDT_POS_STE:
'*****

RETURN

'*****
STARTUP:
'*****
    GOSUB INIT_ADC             'Initialize the ADC
    GOSUB INIT_MOTORS         'Initialize the Motors
RETURN

'*****
INIT_ADC:
'*****
    HIGH ADC_PIN_CS           'Deactivate ADC to begin
    HIGH ADC_PIN_DAT          'Set data pin for first start bit
RETURN

'*****
INIT_MOTORS:
'*****

'Reset the PAK
OUTPUT PAK_PIN_CLK           'Set PAK Clock Pin to Output
OUTPUT PAK_PIN_DAT           'Set PAK Data Pin to Output
GOSUB PAK_RESET              'Reset the PAK-VIII
PAUSE 1000                   'Wait for a second

'Init Motor Direction Parameters
AZI_DIR_CUR = 0               'Set Azimuth Motor Direction to Left (0)
ELE_DIR_CUR = 0               'Set Elevation Motor Direction to Down (0)
AZI_SPD_CUR = 1               'Set Azimuth Motor Speed to 1
ELE_SPD_CUR = 1               'Set Elevation Motor Speed to 1
AZI_SPD_LST = 1               'Set the Last Speed Sent to PAK to 1
ELE_SPD_LST = 1               'Set the Last Speed Sent to PAK to 1

'Force Logic Lows on the Direction Pulse Channels
PAK_VAR_CHN = AZI_PIN_DIR     'Set Azimuth Motor Initial Direction
TMP_BYTE = AZI_DIR_CUR        'Set Initial Direction = Left
GOSUB PAK_SET_PIN             'Make it so

PAK_VAR_CHN = ELE_PIN_DIR     'Set Elevation Motor Initial Direction
TMP_BYTE = ELE_DIR_CUR        'Set Initial Direction = Down
GOSUB PAK_SET_PIN             'Make it so

'Load Default Pulses and Turn Speed Control Channels On for Each Motor
PAK_VAR_CHN = AZI_PIN_SPD     'Set the channel to Azimuth

PAK_VAR_REG = 1               'Set the On Duration Register
TMP_WORD = AZI_SPD_CUR        'Set value to Azimuth Speed (1 = default)
GOSUB PAK_BLD_CMD             'Send the update

PAK_VAR_REG = 2               'Set the Off Duration Register
TMP_WORD = 50 - AZI_SPD_CUR   'Set value to 50 - Azimuth Speed (49 = default)
GOSUB PAK_BLD_CMD             'Send the update

PAK_VAR_REG = 6               'Set the Enable Register
TMP_WORD = 0                  'Enable this Channel
GOSUB PAK_BLD_CMD             'Send the update

PAK_VAR_CHN = ELE_PIN_SPD     'Set the channel to Elevation

PAK_VAR_REG = 1               'Set the On Duration Register
TMP_WORD = ELE_SPD_CUR        'Set value TO Elevation Speed (1 = default)
GOSUB PAK_BLD_CMD             'Send the update

PAK_VAR_REG = 2               'Set the Off Duration Register

```



```

RETURN
'*****
SET_VALID_POS:
'*****

SELECT MTR_NUM_CUR

CASE 0
'Azimuth Motor

IF (CTL_CMD_DAT(1) <= AZI_POS_MAX) AND (CTL_CMD_DAT(1) >= AZI_POS_MIN) THEN 'If Valid - Set the Current Target
  AZI_POS_TGT = CTL_CMD_DAT(1) 'Otherwise leave the last good value
ENDIF

CASE 1
'Elevation Motor

IF (CTL_CMD_DAT(2) <= ELE_POS_MAX) AND (CTL_CMD_DAT(2) >= ELE_POS_MIN) THEN
  ELE_POS_TGT = CTL_CMD_DAT(2)
  'SEROUT 16,16624, ["Target Position: ",DEC ELE_POS_TGT, CR]

ENDIF

ENDSELECT

RETURN
'*****
SET_CUR_MTR_STATE:
'*****

'SEROUT 16,16624, ["Loading State for Motor ",DEC MTR_NUM_CUR, "...", CR]

SELECT MTR_NUM_CUR

CASE 0

MTR_PIN_DIR = AZI_PIN_DIR
MTR_PIN_SPD = AZI_PIN_SPD
MTR_POS_TGT = AZI_POS_TGT
MTR_POS_MIN = AZI_POS_MIN
MTR_POS_MAX = AZI_POS_MAX
MTR_SPD_CUR = AZI_SPD_CUR
MTR_SPD_LST = AZI_SPD_LST
MTR_SPD_TGT = AZI_SPD_TGT
MTR_DIR_CUR = AZI_DIR_CUR

CASE 1

MTR_PIN_DIR = ELE_PIN_DIR
MTR_PIN_SPD = ELE_PIN_SPD
MTR_POS_TGT = ELE_POS_TGT
MTR_POS_MIN = ELE_POS_MIN
MTR_POS_MAX = ELE_POS_MAX
MTR_SPD_CUR = ELE_SPD_CUR
MTR_SPD_LST = ELE_SPD_LST
MTR_SPD_TGT = ELE_SPD_TGT
MTR_DIR_CUR = ELE_DIR_CUR

ENDSELECT

RETURN
'*****
SYNCH_POS:
'*****

SELECT MTR_NUM_CUR

CASE 0

CTL_CMD_DAT(1) = MTR_POS_CUR

CASE 1

CTL_CMD_DAT(2) = MTR_POS_CUR

ENDSELECT

RETURN
'*****
SYNCH_VARS:
'*****

SELECT MTR_NUM_CUR

CASE 0

AZI_SPD_CUR = MTR_SPD_CUR
AZI_SPD_LST = MTR_SPD_LST
AZI_SPD_TGT = MTR_SPD_TGT
AZI_DIR_CUR = MTR_DIR_CUR
CTL_CMD_DAT(1) = MTR_POS_CUR

CASE 1

ELE_SPD_CUR = MTR_SPD_CUR
ELE_SPD_LST = MTR_SPD_LST
ELE_SPD_TGT = MTR_SPD_TGT
ELE_DIR_CUR = MTR_DIR_CUR
CTL_CMD_DAT(2) = MTR_POS_CUR

ENDSELECT

RETURN
'*****
READ_POSITION_SENSOR:
'*****
'Main ADC Loop

```

```
ADC_CFG_DAT = ADC_CFG_DAT | %1011          'Set all ADC Config Bits Except Channel
LOW_ADC_PIN_CS                             'Activate the ADC
SHIFTOUT ADC_PIN_DAT,ADC_PIN_CLK,LSBFIRST,[ADC_CFG_DAT\4]'Send ADC Config Bits
SHIFTLIN ADC_PIN_DAT,ADC_PIN_CLK,MSBPOST,[TMP_WORD\12] 'Get data bits
HIGH_ADC_PIN_CS                             'Deactivate the ADC
MTR_POS_CUR = TMP_WORD/16                   'Return the (compressed) Results into the Current Motor State
'SEROUT 16,16624, ["Current Motor: ", DEC MTR_NUM_CUR, CR, "Current Position: ", DEC MTR_POS_CUR, CR]
RETURN

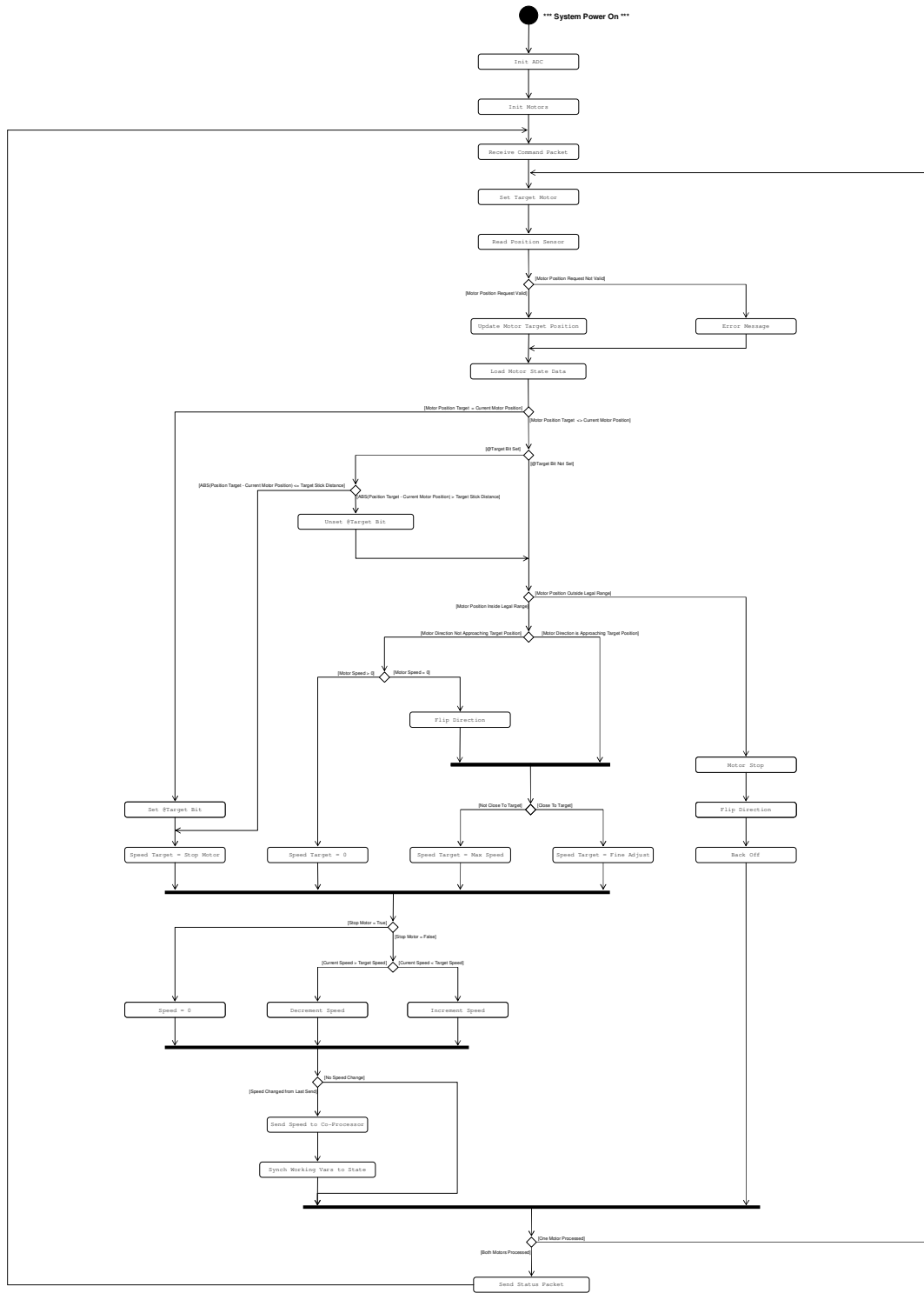
'*****
PAK_RESET:
'*****
LOW_PAK_PIN_DAT
LOW_PAK_PIN_CLK
HIGH_PAK_PIN_CLK
HIGH_PAK_PIN_DAT
LOW_PAK_PIN_CLK
RETURN

'*****
PAK_SET_PIN:
'*****
' Pass Tmp_Byte as argument with either a 0 or 1 set for logic high or low
' 0 is LOW, 1 is HIGH, 2 is hi-z, 3 is normal
' note: 3 enables channel to run!
TMP_BYTE = PAK_VAR_CHN + (TMP_BYTE<<3)+%01000000
GOTO PAK_SND_CMD

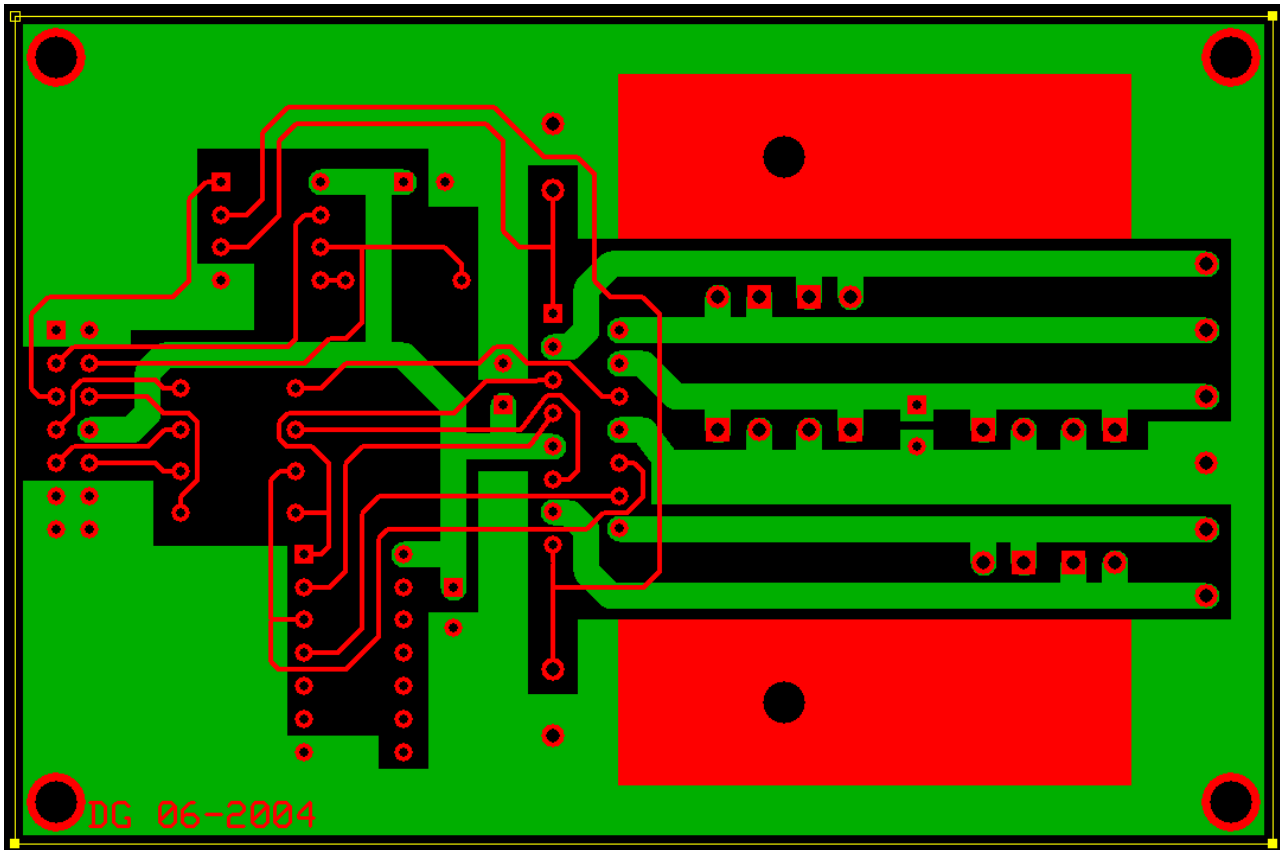
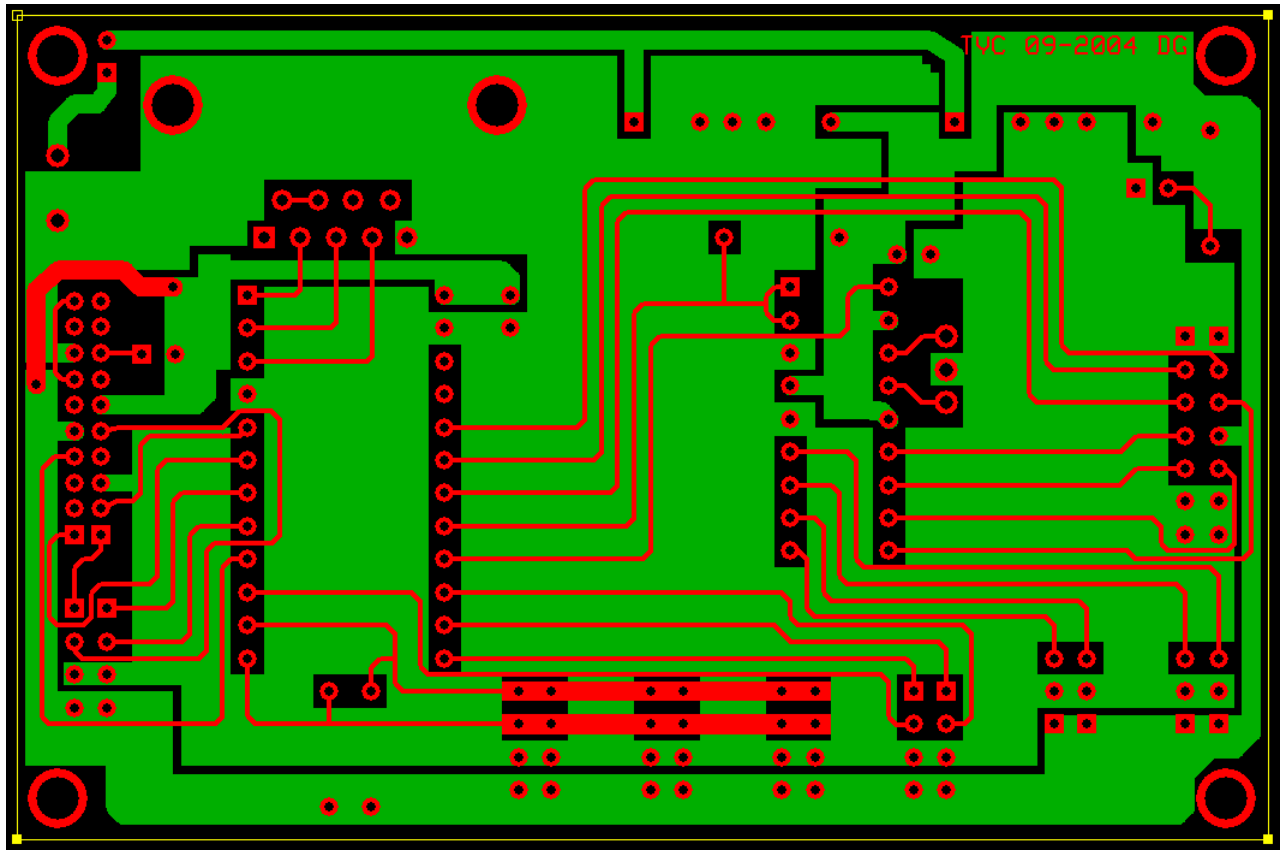
'*****
PAK_BLD_CMD:
'*****
TMP_BYTE = (PAK_VAR_REG<<3) + PAK_VAR_CHN
GOSUB PAK_SND_CMD
TMP_BYTE = TMP_WORD.LOWBYTE
GOSUB PAK_SND_CMD
TMP_BYTE = TMP_WORD.HIGHBYTE

'*****
PAK_SND_CMD:
'*****
SHIFTOUT PAK_PIN_DAT,PAK_PIN_CLK,MSBFIRST,[TMP_BYTE]
RETURN
```

APPENDIX B - ANTENNA SCAN CONTROLLER FIRMWARE LOGIC DIAGRAM



APPENDIX C - ANTENNA SCAN CONTROLLER PCB PATTERNS



APPENDIX D - SAMPLE ANTENNA SCAN DRIVER SOURCE

```

#include <stdio.h> /* Standard input/output definitions */
#include <string.h> /* String function definitions */
#include <unistd.h> /* UNIX standard function definitions */
#include <fcntl.h> /* File control definitions */
#include <errno.h> /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */
#include <sys/ioctl.h>

int fd;

int initport(int fd) {

    struct termios options;
    // Get the current options for the port...
    tcgetattr(fd, &options);
    // Set the baud rates to 19200...
    cfsetispeed(&options, B4800);
    cfsetospeed(&options, B4800);
    // Enable the receiver and set local mode...
    options.c_cflag |= (CLOCAL | CREAD);

    options.c_cflag &= ~PARENB;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;
    //options.c_lflag |= ~(ICANON);
    options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
    options.c_iflag &= ~(IXON | IXOFF | IXANY);
    options.c_oflag &= ~OPOST;
    // Set the new options for the port...
    tcsetattr(fd, TCSANOW, &options);

    return 1;
}

int main(int argc, char **argv) {

    fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
    if (fd == -1) {
        perror("open_port: Unable to open /dev/ttyS0 - ");
        return 1;
    } else {
        fcntl(fd, F_SETFL, 0);
    }
    initport(fd);

//Drop the DTR Line
    int status;

    ioctl(fd, TIOCMGET, &status);

    status &= ~TIOCM_DTR;

    ioctl(fd, TIOCMSET, &status);

    char sCmd[254];

    sCmd[0] = 0x46;
    sCmd[1] = 0x59;
    sCmd[2] = 0x80;
    sCmd[3] = 0x80;
    sCmd[4] = 0x00;

    int cntr;
    int lc;
    int bytes;
    char sResult[254];

    //tcflush(fd, TCIOFLUSH);

    //usleep(500000);

    for(lc = 0;lc < 1000;lc++)
    {

        printf("written:%s\n", sCmd);

        if (!writeport(fd, sCmd)) {
            printf("write failed\n");
            close(fd);
            return 1;
        }

        //usleep(200000);

        while (bytes < 8)
        {
            ioctl(fd, FIONREAD, &bytes);
            usleep(10000);
            printf("RX Bytes: %d\n",bytes);
        }

        if (!readport(fd,sResult)) {
            printf("read failed\n");
            close(fd);
            return 1;
        }
        printf("readport=%s\n", sResult);
        //tcflush(fd, TCIFLUSH);
        tcflush(fd, TCIFLUSH);
        usleep(50000);
    }
    close(fd);
}

```

```
    return 0;
}

int writeport(int fd, char *chars) {
    int n = write(fd, chars, 4);
    if (n < 0) {
        fputs("write failed!\n", stderr);
        return 0;
    }
    return 1;
}

int readport(int fd, char *result) {
    int iIn = read(fd, result, 254);
    result[iIn-1] = 0x00;

    printf("Read %d bytes...\n", iIn);

    if (iIn < 0) {
        if (errno == EAGAIN) {
            printf("SERIAL EAGAIN ERROR\n");
            return 0;
        } else {
            printf("SERIAL read error %d %s\n", errno, strerror(errno));
            return 0;
        }
    }
    return 1;
}

int getbaud(int fd) {
    struct termios termAttr;
    int inputSpeed = -1;
    speed_t baudRate;
    tcgetattr(fd, &termAttr);
    /* Get the input speed. */
    baudRate = cfgetispeed(&termAttr);
    switch (baudRate) {
        case B0:      inputSpeed = 0; break;
        case B50:     inputSpeed = 50; break;
        case B110:    inputSpeed = 110; break;
        case B134:    inputSpeed = 134; break;
        case B150:    inputSpeed = 150; break;
        case B200:    inputSpeed = 200; break;
        case B300:    inputSpeed = 300; break;
        case B600:    inputSpeed = 600; break;
        case B1200:   inputSpeed = 1200; break;
        case B1800:   inputSpeed = 1800; break;
        case B2400:   inputSpeed = 2400; break;
        case B4800:   inputSpeed = 4800; break;
        case B9600:   inputSpeed = 9600; break;
        case B19200:  inputSpeed = 19200; break;
        case B38400:  inputSpeed = 38400; break;
    }
    return inputSpeed;
}
```