

Android Based Antenna Positioning System

Girish Mungi, L. Srinija, M. Navyanth

Department of Electronics and Communication Engineering
Nalla Narasimha Reddy Education Society's Group of Institutions
Hyderabad, India

Abstract - This paper describes a laboratory antenna-positioning model operated from an Android TCP Telnet Terminal application over a local Wi-Fi connection. The implementation combines an Arduino-based controller, a serial Wi-Fi interface, two DC motor channels, an ADXL345 accelerometer, and a local liquid-crystal display. Character commands request movement along the two model axes; each movement command energizes one motor channel for a programmed 600 ms interval. The firmware estimates roll and pitch from acceleration measurements and returns signed X and Y values to the terminal. Prototype photographs, an application screenshot, and the firmware listing document the implementation. The system is an operator-directed positioning demonstrator with tilt telemetry, rather than an automatic target-angle controller. This paper details the command protocol, pin assignments, sensing calculations, and timing sequence, and distinguishes observed terminal output from calibrated angular measurements. Quantitative pointing accuracy, communication range, and radio-frequency reception improvement remain to be characterized.

Keywords - *antenna positioning; Android; Wi-Fi; TCP terminal; DC motor; ADXL345; tilt sensing*

I. INTRODUCTION

A motorized antenna mount allows an operator to change orientation without adjusting the assembly by hand. A mobile interface can supply directional commands while a controller drives the motors and displays sensor information. These functions are useful in a laboratory demonstrator because communication, actuation, and sensing can be examined within one apparatus.

The present project uses an Android TCP terminal to operate two motor channels, identified as the model's X and Y directions. It combines directional motion commands with accelerometer-derived tilt telemetry. This distinction is central to the implementation: the operator requests a direction, not a numerical target angle. The firmware applies a fixed-duration motor pulse and reports sensor values; it does not compare a desired angle with a measured angle to stop at a target.

This paper describes the hardware, control sequence, and observations from the laboratory model. Its scope is the implemented motion and monitoring arrangement; calibrated pointing performance and radio-frequency reception require further evaluation.

II. RELATED WORK

Tay Mei Lin, Maisarah Abu, and Siti Adlina Md Ali [1] reported an Android-based antenna positioning arrangement using a NodeMCU controller, stepper motors, orientation sensing, and a mobile interface. Their study included radio-frequency measurements. Revane et al. [2] described an IoT

antenna positioning arrangement combining an ATmega328 controller, accelerometer feedback, motor actuation, and a networked interface. These studies establish prior work in remotely operated antenna positioning.

The present implementation integrates two-channel DC motor control, TCP terminal messages, a local display, and ADXL345 calculations. This paper documents their operation in a laboratory model without claiming a new positioning algorithm or comparative performance advantage.

III. HARDWARE AND SYSTEM ORGANIZATION

A. Controller and actuator arrangement

The project schematic identifies an Arduino Nano controller and an L293D motor-drive stage. The classic Nano provides the digital and analog interfaces used by this design [3]. Two DC motors provide motion along the two model directions. The L293D supplies half-H driver stages suitable for inductive loads [4]; the controller determines direction through two digital outputs per motor. Figure 1 shows the assembled model, local display, interconnected boards, drive mechanism, and sensor board attached to the movable structure.

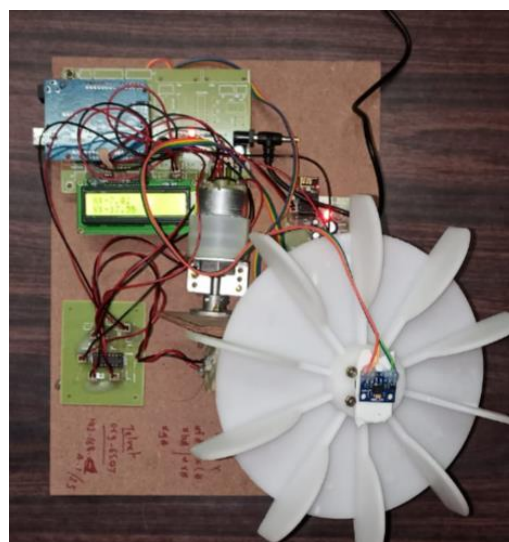


Fig. 1. Laboratory model with the display, drive mechanism, and sensor board visible.

The firmware initializes all four motor outputs LOW. For a directional command, one output of the selected pair is driven HIGH while the other remains LOW. After 600 ms, both outputs return LOW. This pulse-based operation controls movement duration; the physical angular displacement also depends on motor speed, loading, gearing, and supply conditions. The two motor channels are designated M1 and M2 here because a calibrated mapping to antenna azimuth and elevation has not been established.

B. Sensor and local display

The firmware addresses an ADXL345 digital accelerometer at I2C address 0x53. It configures the sensor for a nominal 100 Hz output data rate, measurement mode, and full-resolution operation at the ± 2 g range. The ADXL345 measures acceleration, including static gravity used for tilt estimation [5]. It is not a magnetic heading sensor. The firmware identification supersedes the generic magnetometer label in the earlier project schematic.

A 16×2 character display presents X and Y values locally. The LiquidCrystal constructor assigns control pins D6 and D7 and data pins D5, D4, D3, and D2. Serial communication with the Wi-Fi interface is initialized at 9600 baud. Table I records the assignments explicitly present in the source listing; it is not a reconstruction of every wire or supply connection.

TABLE I. INTERFACE ASSIGNMENTS FROM THE FIRMWARE

Function	Firmware assignment
Motor M1	D8 and D9
Motor M2	D10 and D11
LCD control	RS D6; enable D7
LCD data	D5, D4, D3, D2
ADXL345	I2C address 0x53 via Wire
Wireless interface	Hardware serial at 9600 baud

IV. ANDROID INTERFACE AND CONTROL FIRMWARE

A. Wi-Fi and TCP session



Fig. 2. TCP Telnet Terminal session showing the endpoint, X/Y text, and command buttons.

The Android application is TCP Telnet Terminal. The recorded session shows the endpoint 192.168.4.1:23, consistent with a local TCP connection. The firmware sends AT+CWMODE=3, AT+CIPMUX=1, and AT+CIPSERVER=1,23 to the Wi-Fi module. These commands select the station/access-point mode, enable multiple connections, and create a TCP server on port 23 in the ESP8266 AT command family [6]. The code therefore documents a Wi-

Fi/TCP path rather than a Bluetooth link. No internet service or cloud relay is required by the shown local session.

The terminal provides configurable ASCII command buttons and a received-text window (Fig. 2). The firmware uses a small application protocol over the TCP stream; a complete Telnet option-negotiation implementation is not present in the listing. It waits for connection-related serial output during setup and then processes command characters received through the serial interface.

B. Command framing and motor actions

Commands are framed by an asterisk and a hash character. The serial-event handler collects characters after “*”, marks a frame complete at “#”, and takes the second character as the command. The main loop recognizes f, b, l, r, and g. Table II gives the resulting output states. The directional names indicate opposite drive polarities; the actual mechanical direction depends on motor wiring and mounting.

TABLE II. COMMANDS IMPLEMENTED IN THE SOURCE LISTING

Frame	Selected outputs	Action
*f#	D8 HIGH; D9 LOW	M1 pulse
*b#	D8 LOW; D9 HIGH	M1 reverse pulse
*l#	D10 HIGH; D11 LOW	M2 pulse
*r#	D10 LOW; D11 HIGH	M2 reverse pulse
*g#	No motor change	Send X/Y values

Each of the four movement branches holds its drive state for 600 ms, stops the selected motor, and calls the telemetry function. The g branch sends telemetry without a motor pulse. There is no numerical angle setpoint, proportional control law, stepper sequence, or continuous position-error correction in these branches. The apparatus is therefore described as operator-directed motion with sensor monitoring.

C. Tilt calculation and displayed values

The sensor routine reads six bytes from registers 0x32 through 0x37 and reconstructs signed acceleration counts ax, ay, and az. It calculates roll and pitch using the following expressions, with the factor 57.3 converting radians approximately to degrees:

$$\text{roll} = 57.3 \operatorname{atan2}(\text{ay}, \text{az})$$

$$\text{pitch} = 57.3 \operatorname{atan2}(-\text{ax}, \sqrt{\text{ay}^2 + \text{az}^2})$$

The code then adds 5 to roll when roll is below -40 and subtracts 5 when roll is positive. These conditional offsets are part of the implementation, but their calibration basis is not documented. X is assigned the adjusted roll and Y the pitch. The local display prints the floating-point values, whereas the terminal function converts them to integers and transmits a sign and three digits for each coordinate.

These quantities estimate orientation relative to gravity under suitable static conditions. They are not independently measured antenna angles. Motor vibration or acceleration can affect the readings, and gravity alone does not establish compass heading. In particular, a mount axis that rotates only about the vertical cannot be characterized as a measured azimuth solely from these calculations.

D. Sampling and communication sequence

Each loop reads the accelerometer, refreshes the LCD, and waits 200 ms before processing a completed command. The telemetry routine requests a 15-byte send on connection 0, waits

2 s, transmits the formatted X/Y line, and waits another 3 s. These are programmed delays, not measured response times. Together with the 600 ms pulse, they prevent interpreting the sensor's configured 100 Hz rate as the application update rate.

For a movement command, the sample used by the telemetry routine was acquired before the motor pulse in the same loop. There is no new accelerometer read between stopping the motor and sending that message. Consequently, the first returned line after a movement command may describe the earlier pose. A later sample, such as one obtained before a subsequent g request, is needed to observe a post-movement estimate. This sequence is a limitation of the existing firmware, not a claim that a corrected version was tested.

V. PROTOTYPE OBSERVATIONS AND LIMITATIONS

A. Available demonstration record

The project team reports operation in both X and Y directions. The photographs show the assembled laboratory model with its local display illuminated. The TCP terminal screenshot documents a session containing several signed coordinate pairs, including X +034 with Y -001 and X +007 with Y -012. These are visible device-reported values. The screenshot does not associate each value with a specified command, reference angle, or timed trial, so it cannot establish positioning error or repeatability.

The physical assembly and terminal record support the presentation of an implemented control interface and reported two-axis demonstration. The firmware listing supplies the motor actions, sensor computations, and packet format. This evidence is useful for understanding the implementation, but it is not a calibrated performance dataset. No numerical claims are made for angular accuracy, operating distance, settling time, or received-signal improvement.

B. Firmware and measurement limitations

The blocking motor and communication delays reduce responsiveness to later commands. The parser stores a completed command in a single variable rather than a queue, so it is not designed to preserve every command in a rapid burst. The setup checks single characters in serial output rather than validating complete responses with timeouts. Telemetry is sent to connection 0, although multiple connections are enabled. These choices suit a simple demonstration but require revision for reliable unattended operation.

The transmitted X/Y message is 15 bytes with the stated punctuation and line ending. The screenshot uses a closely related presentation, and no archived executable is available to confirm the exact firmware revision running when it was captured. The source listing also lacks motion-limit handling and a calibrated conversion between motor duration and mechanical displacement. The apparatus should therefore be evaluated as a laboratory positioning model, without assuming field-ready autonomous operation.

C. Further experimental characterization

A quantitative assessment would record each command, initial pose, independently measured final pose, and movement direction under fixed supply and load conditions. Repeated trials in both directions would reveal displacement variability and

mechanical backlash. Timing should distinguish command-to-motion delay, motor-on interval, settling time, and telemetry arrival. The 600 ms constant must remain separate from all measured timing results.

Communication testing would record successful and failed commands at known distances and under stated network conditions. A reception study would additionally require a real antenna, operating frequency, transmitter arrangement, receiving instrument, and comparable signal measurements across orientations. These tests are proposed future work; they were not reconstructed from the terminal screenshot or borrowed from published studies.

VI. CONCLUSION

The Android-based model integrates a Wi-Fi TCP terminal, two DC motor channels, ADXL345 tilt calculations, and a local display. The implementation supports directional motor pulses and signed X/Y telemetry. Its firmware establishes the command mapping and timing sequence, while the photographs and application record document the laboratory arrangement. The current system provides operator-directed movement and orientation estimates; it does not implement calibrated target-angle control. Further work should address fresh post-movement sampling, communication handling, sensor calibration, and independent mechanical and radio-frequency testing before stronger performance claims are made.

REFERENCES

- [1] Tay Mei Lin, Maisarah Abu, and Siti Adlina Md Ali, "Android based antenna positioning system," *Jurnal Teknologi*, vol. 84, no. 3, pp. 81–88, 2022, doi: 10.11113/jurnalteknologi.v84.17287.
- [2] P. Revane, S. Salaskar, K. Shelke, P. Tawar, and A. Raut, "Implementing an IOT based antenna positioning system," *International Journal for Research in Applied Science & Engineering Technology*, vol. 6, no. IV, pp. 2898–2900, Apr. 2018, doi: 10.22214/ijraset.2018.4483.
- [3] Arduino, "Nano," hardware documentation. [Online]. Available: <https://docs.arduino.cc/hardware/nano>. Accessed: Sep. 25, 2026.
- [4] Texas Instruments, "L293x quadruple half-H drivers," SLRS008D, rev. D, Jan. 2016. [Online]. Available: <https://www.ti.com/lit/gpn/L293D>.
- [5] Analog Devices, "ADXL345: 3-axis, ± 2 g/ ± 4 g/ ± 8 g/ ± 16 g digital accelerometer," data sheet, rev. G, 2022. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf>.
- [6] Espressif Systems, "ESP8266 AT command examples," version 1.3. [Online]. Available: https://www.espressif.com/sites/default/files/4b-esp8266_at_command_examples_en_v1.3.pdf.