MockSat v3.1

PRISM Capstone

Complete User and Operator Guide

Proximity and Rendezvous Integrated Simulation Model

Platform	Teensy 4.1
Firmware	MockSat v3.1
Display	LCD1602 16x2 I2C (0x27)
IMU	Adafruit BNO055 (I2C 0x28)
Bluetooth	HM-10 BLE Module (Serial1)
Motor Channels	8 (ESC-controlled thrusters)
Author	Ryan Coder
Document Type	User and Operator Guide

1. System Overview

MockSat v3.1 is an embedded firmware application for the Teensy 4.1 microcontroller, developed as part of the PRISM Capstone project (Proximity and Rendezvous Integrated Simulation Model). It simulates the onboard computer of a small proximity operations satellite, providing a complete LCD menu interface, real-time inertial measurement using the BNO055 sensor, wireless position telemetry via the HM-10 Bluetooth Low Energy module, and a motor throttle abstraction layer for up to eight thruster channels.

The system is intended to operate alongside a Ground Truth and Simulation System (GTSS). The GTSS sends commanded target positions (CMD) and current position telemetry (TEL) to the MockSat over BLE. The MockSat displays both values alongside its own locally estimated position, enabling operators to monitor navigation accuracy in real time.

1.1 Hardware Summary

Component	Part / Description	Interface	Address or Pin
Microcontroller	PJRC Teensy 4.1	USB / Native	N/A
LCD Display	LCD1602 with I2C backpack	I2C (Wire)	0x27
IMU	Adafruit BNO055	I2C (Wire)	0x28
Bluetooth Module	HM-10 BLE	UART (Serial1)	9600 baud
Button K1	Select / Confirm	Digital Input	Pin 14
Button K2	Up / Increase	Digital Input	Pin 15
Button K3	Down / Decrease	Digital Input	Pin 16
Button K4	Back / Cancel	Digital Input	Pin 17
Motors (x8)	ESC-driven thrusters	PWM	HiLetgo PWM board

1.2 Software Dependencies

Library	Purpose	Install Source
Wire	I2C bus communication (built-in)	Arduino / Teensyduino built-in
LiquidCrystal_I2C	LCD1602 I2C display driver	Arduino Library Manager
Adafruit_Sensor	Unified sensor abstraction layer	Arduino Library Manager
Adafruit_BNO055	BNO055 IMU driver	Arduino Library Manager
utility/imumaths.h	IMU math utilities (bundled with BNO)	Included with Adafruit_BNO055

1.3 Key Capabilities

- Animated multi-step boot sequence with per-subsystem initialization display and live IMU calibration scoring
- Hierarchical 16x2 LCD menu system navigable entirely with four physical push buttons
- Real-time BNO055 sensor polling at a configurable interval covering Euler angles, gyroscope rates, linear and total acceleration, magnetometer field strength, gravity vector, and chip temperature
- Bluetooth Low Energy receive channel accepting TEL (current position) and CMD (commanded target) messages from the GTSS in the format x,y,theta
- HM-10 AT command interface for switching between slave and master BLE roles, scanning for nearby devices, and initiating connections by MAC address
- Eight-channel motor throttle state table with ESC arming logic and mode-driven throttle assignment
- Eight selectable operating modes including autonomous point navigation stub, Xbox controller passthrough stub, idle, full throttle, half throttle, and positive and negative Z-axis spin profiles
- Configurable persistent settings for default throttle percentage, motor test duration in seconds, and IMU update rate in milliseconds
- Full diagnostics panel covering five pages of IMU sensor data, temperature, battery (placeholder), all eight motor throttle values, estimated local position versus received telemetry, and GTSS commanded position versus telemetry
- Soft system reset with full state clearing accessible from the main menu

2. Hardware Wiring and Assembly

All wiring must be completed and verified before powering the system or uploading firmware. Pay close attention to voltage levels. The HM-10 module uses 3.3V logic; the Teensy 4.1 GPIO is also 3.3V, making them directly compatible. However, the power supply pin on the HM-10 must be connected carefully as some breakout boards accept 5V on VCC while others do not.

2.1 LCD1602 I2C Display

The LCD uses a PCF8574-based I2C backpack at address 0x27. It shares the I2C bus with the BNO055. Both devices can coexist on the same SDA/SCL lines because they have different addresses.

LCD1602 Pin	Connects To	Notes
VCC	Teensy 5V or 3.3V	Most I2C LCD backpacks accept 5V; check your module datasheet
GND	Teensy GND	Must share ground with Teensy
SDA	Teensy Pin 18	I2C data line, shared with BNO055
SCL	Teensy Pin 19	I2C clock line, shared with BNO055

2.2 BNO055 IMU

The BNO055 is a 9-degree-of-freedom inertial measurement unit with an integrated sensor fusion processor. It is addressed at 0x28 by default when the ADR pin is tied to ground. The firmware uses the external crystal oscillator mode for improved timing accuracy.

BNO055 Pin	Connects To	Notes
VIN / VCC	Teensy 3.3V	BNO055 operates at 3.3V
GND	Teensy GND	Shared ground
SDA	Teensy Pin 18	Shared I2C data line
SCL	Teensy Pin 19	Shared I2C clock line
ADR	GND	Pulls I2C address to 0x28 (default). Leave floating or tie to 3.3V for 0x29

NOTE If your BNO055 module has its address configured to 0x29 rather than 0x28 via a solder bridge or jumper, you must update the constructor in the source code from Adafruit_BNO055(55, 0x28, &Wire) to Adafruit_BNO055(55, 0x29, &Wire) before uploading.

2.3 HM-10 BLE Module

The HM-10 is a Texas Instruments CC2541-based Bluetooth Low Energy module. It communicates with the Teensy over a 3.3V UART at 9600 baud. The firmware uses Serial1, which maps to Teensy pins 0 (RX) and 1 (TX).

HM-10 Pin	Connects To	Notes
VCC	Teensy 3.3V	CRITICAL: Do not connect to 5V if your module has no on-board regulator
GND	Teensy GND	Shared ground
TXD	Teensy Pin 0 (RX1)	HM-10 transmits data to Teensy receive line
RXD	Teensy Pin 1 (TX1)	Teensy transmits AT commands to HM-10 receive line

WARNING The HM-10 module logic is 3.3V. Teensy 4.1 GPIO is 3.3V. No level shifter is required. However, if you are using a 5V Arduino or breakout board adapter, a level shifter or voltage divider is mandatory on the RXD line to avoid damaging the module.

2.4 Button Wiring

All four buttons use internal pull-up resistors activated via INPUT_PULLUP in firmware. No external resistors are needed. Each button connects between its assigned Teensy pin and GND. When pressed, the pin reads LOW, which the firmware treats as a press event.

Button ID	Firmware Function	Teensy Pin	Wiring
K1	Select / Confirm	14	One leg to pin 14, other leg to GND
K2	Up / Increase	15	One leg to pin 15, other leg to GND
K3	Down / Decrease	16	One leg to pin 16, other leg to GND
K4	Back / Cancel	17	One leg to pin 17, other leg to GND

NOTE All four buttons can share a single common GND bus. Run one wire from any Teensy GND pin to a breadboard ground rail, then wire each button from its signal pin to that rail.

2.5 Motor ESC Wiring

The firmware defines eight motor channels (M1 through M8) and maintains a throttle state array. The actual PWM output to ESCs is currently marked as a TODO in the source code (see setMotorThrottle). The intended path is through a HiLetgo PCA9685 16-channel PWM driver board connected over I2C. Motor wiring will depend on your specific ESC and thruster configuration. The firmware currently logs all throttle commands to the Serial Monitor so you can verify behavior before connecting physical ESCs.

WARNING Do not connect powered ESCs to the system during software development and testing phases. The firmware will ARM the ESC array during boot. An armed ESC connected to a powered motor can begin spinning if any throttle value is set non-zero.

3. Software Installation and Firmware Upload

3.1 Install Arduino IDE and Teensyduino

1. Download Arduino IDE 2.x from https://www.arduino.cc/en/software and run the installer for your operating system.
2. Download the Teensyduino add-on from https://www.pjrc.com/teensy/teensyduino.html and run the installer. Point it to your Arduino IDE installation directory when prompted.
3. Launch Arduino IDE. Navigate to Tools > Board > Teensyduino and select Teensy 4.1.
4. Navigate to Tools > USB Type and select Serial. This enables the USB Serial port used for the debug monitor.
5. Leave CPU Speed at the default 600 MHz.

3.2 Install Required Libraries

Open the Library Manager via Sketch > Include Library > Manage Libraries. Search for and install each of the following:

Search Term	Library to Install	Notes
LiquidCrystal I2C	LiquidCrystal_I2C by Frank de Brabander	Version 1.1.2 or later
Adafruit BNO055	Adafruit BNO055 by Adafruit	Install all dependencies when prompted
Adafruit Unified	Adafruit Unified Sensor by Adafruit	Required dependency for BNO055 driver

NOTE When installing the Adafruit BNO055 library, the Library Manager will prompt you to install its dependencies. Click Install All to ensure Adafruit Unified Sensor is also installed. If you click Only Install, the firmware will fail to compile.

3.3 Open and Upload the Firmware

6. Open the MockSat .ino file in Arduino IDE via File > Open.
7. Verify the sketch compiles without errors using Sketch > Verify/Compile (Ctrl+R). Resolve any missing library errors before proceeding.
8. Connect the Teensy 4.1 to your computer via USB.
9. The correct port should appear automatically under Tools > Port. Select it.
10. Click Upload (Ctrl+U). The Teensy Loader application will launch automatically and flash the firmware.
11. After uploading, the LCD will illuminate and the boot sequence will begin within one second.

NOTE If the Teensy does not appear in the port list, press the small physical programming button on the Teensy 4.1 board. This forces it into bootloader mode and makes it visible to the Teensy Loader.

3.4 Serial Monitor

The firmware outputs extensive diagnostic information over USB Serial at 115200 baud. To view it, open Tools > Serial Monitor in Arduino IDE and set the baud rate to 115200. You will see the full startup banner, all button press events labeled by button ID, all incoming Bluetooth messages with raw content, motor throttle assignments, IMU calibration progress, AT command transmissions, and all menu state transitions. The Serial Monitor is the primary debug tool and should remain open during any development or troubleshooting session.

Sample Serial Monitor Output

```
    +----------------------------------------------------------+
    |              PRISM  MockSat  v3.1                        |
    |      Proximity & Rendezvous Integrated Simulation        |
    |                   Author: Ryan Coder                     |
    |                  Platform: Teensy 4.1                    |
    +----------------------------------------------------------+


[BOOT] MockSat v3.1 Initializing...
[BOOT] Bluetooth HM-10: Serial1 @ 9600 baud
[BOOT] I2C Wire: OK
[BOOT] IMU BNO055: OK
[INPUT] K2 - Up
[BT] <<< Raw: TEL:1.5,3.2,90.0
[TEL] Received position -> x: 1.50  y: 3.20  theta: 90.00
```

4. Boot Sequence

On every power-up or after a system reset via the Reset menu option, the MockSat runs a timed multi-step initialization sequence. Each step initializes a hardware subsystem and displays its status on the LCD. The boot sequence cannot be skipped. Under normal conditions with an IMU present, it completes in approximately 15 to 60 seconds, with most of that time spent on IMU calibration.

Step	LCD Line 1	LCD Line 2	Duration	Description
0	MockSat v3.1	Initializing...	Immediate	Firmware version displayed on power-up
1	Init Bluetooth	HM-10 Serial1	1.5 s	Serial1 UART opened at 9600 baud
2	Init I2C Wire	Status: OK	1.0 s	I2C bus initialized; LCD and IMU become accessible
3	Init IMU	Status: OK / N/A	1.0 s	BNO055 detection attempted at 0x28
3b	Calibrating IMU	S:x G:x M:x	Variable	Live calibration scores shown; blocks until all reach 3
3c	IMU Calibrated	Ready!	1.5 s	Zero-reference orientation captured and stored
4	Init Motors	Arming ESCs...	1.0 s	motorThrottle array zeroed; motorsArmed set to true
5	System Ready!	Loading menu...	1.0 s	Boot complete; transitioning to main menu

4.1 IMU Calibration During Boot

The BNO055 does not retain calibration data after power loss. Every time the system boots and the IMU is detected, a calibration procedure must be completed before the boot sequence continues. The LCD displays three scores: S (system), G (gyroscope), and M (magnetometer), each ranging from 0 to 3. Boot will not proceed past this step until S, G, and M all equal 3.

Calibration Procedure by Sensor

- Gyroscope (G): Place the MockSat completely still on a flat surface for several seconds. The gyroscope self-calibrates while motionless.
- Magnetometer (M): Slowly rotate the MockSat through a full figure-eight pattern covering all three axes. Repeat multiple times. Move away from large metallic objects, monitors, and speakers, which create magnetic interference.
- Accelerometer (A): Although not shown on the LCD in this firmware, the accelerometer contributes to the system score. Place the board flat on each of its six faces for two to three seconds each to calibrate it.

- System (S): The system score reaches 3 automatically once the other sensors are sufficiently calibrated. It is the last value to reach 3.

NOTE If the IMU is not physically present or fails to respond on I2C, the boot sequence will display Status: N/A and skip all calibration. The variable imuAvailable will be set to false, and all IMU-dependent diagnostics screens will display Not Available. All other features continue to function normally.

4.2 Calibration Zero-Reference

When calibration completes, the firmware captures the current Euler orientation (X, Y, Z) as a zero-reference offset. All subsequent IMU readings are reported relative to this reference orientation. This means the Diagnostics > IMU Data screen will read approximately 0.00 for all Euler angles immediately after calibration, with values drifting as the unit is physically moved.

The zero-reference is stored in the variables x_offset, y_offset, and z_offset and printed to the Serial Monitor upon calibration completion. These values reset to zero if a system reset is performed and recalibration occurs.

5. Button Controls and Navigation

The entire user interface is driven by four physical push buttons. There is no touchscreen, rotary encoder, or serial command interface for menu navigation. All button inputs use edge detection on the falling edge, meaning a command fires once when the button is pressed down, not continuously while held.

Button	Label	Primary Role	In Adjust Screens	Special Cases
K1	Select / Confirm	Enter highlighted menu item	Save current value and return	In BT Mode: toggles slave/master. In BT Devices: initiates connection.
K2	Up / Increase	Move cursor up in list	Increase value (throttle +5%, test +1s, rate -50ms)	In IMU Diag: previous sensor screen.
K3	Down / Decrease	Move cursor down in list	Decrease value (throttle -5%, test -1s, rate +50ms)	In IMU Diag: next sensor screen.
K4	Back / Cancel	Return to previous menu level	Discard changes and return	At Main Menu: no action. During BT scan: stops scan.

5.1 Navigation Rules

- All list cursors wrap around. Pressing K2 (Up) on the first item jumps to the last item. Pressing K3 (Down) on the last item jumps to the first item. This applies to every list in the system.
- The K4 Back button always returns exactly one level up in the menu hierarchy. There is no multi-level back action.
- Pressing K4 at the Main Menu has no effect. The main menu is the top level.
- Settings adjustment screens require explicit confirmation with K1 to save changes. Using K4 to leave these screens without pressing K1 discards all changes and restores the previous value.
- During a BLE scan, pressing K1 or K4 both stop the scan and navigate to the device list.

5.2 Complete Menu Map

```
MockSat v3.1 Complete Menu Structure
Main Menu
    |-- Pick Mode
    |       |-- PointNav         (stops all motors; navigation stub)
    |       |-- Xbox Control     (stops all motors; controller stub)
    |       |-- Idle             (stops all motors)
    |       |-- Full Throttle    (all 8 motors at 100%)
    |       |-- Half Throttle    (all 8 motors at defaultThrottle %)
    |       |-- Spin +Z          (motors 2,4,6,8 at defaultThrottle %)
    |       |-- Spin -Z          (motors 1,3,5,7 at defaultThrottle %)
    |       +-- Other            (stops all motors; custom stub)
    |
    |-- Bluetooth
    |       |-- View Status      (shows Connected / No Signal + Mode)
    |       |-- Switch Mode      (K1 toggles Slave <-> Master + AT reset)
    |       +-- Scan Devices     (Master only; 6s scan; browse + connect)
    |
    |-- Settings
    |       |-- Default Throttle (K2/K3 adjust in 5% steps; 0-100%)
    |       |-- Motor Test Dur.  (K2/K3 adjust in 1s steps; 1-10s)
    |       |-- IMU Rate         (K2/K3 adjust in 50ms steps; 50-500ms)
    |       +-- Calibrate IMU    (runs CalibrateIMU() immediately)
    |
    |-- Diagnostics
    |       |-- IMU Data         (5 screens: Euler, Gyro, Accel, Mag, Lin)
    |       |-- Temp & Mag       (IMU chip temperature in Celsius)
    |       |-- Battery          (placeholder: Not Available)
    |       |-- Motors           (live M1-M8 throttle values)
    |       |-- Position         (EST local pos vs TEL received pos)
    |       +-- Command          (CMD target pos vs TEL received pos)
    |
    +-- Reset
            +-- Confirm Reset    (K1=Yes performs full soft reset)
```

6. Bluetooth Operation

The MockSat uses an HM-10 BLE module connected to Teensy Serial1 at 9600 baud. The HM-10 firmware supports both a slave role, where it waits for an incoming BLE connection, and a master role, where it can actively scan for and connect to other BLE devices. Role switching is performed by sending AT commands from the MockSat firmware.

6.1 Connection Detection Behavior

The MockSat has no hardware connection-detect pin on the HM-10. The variable bluetoothConnected remains false at all times until the remote system actually transmits a valid TEL or CMD data message over BLE. Simply pairing at the Bluetooth layer, or even connecting at the BLE GAP layer, is not sufficient to set bluetoothConnected to true. The GTSS or remote station must actively send at least one well-formed TEL or CMD message.

Once the first valid message is received, bluetoothConnected is set to true and the Bluetooth > View Status screen will update to show BT: Connected. If the Bluetooth > View Status screen is currently displayed when the first message arrives, it will refresh automatically.

NOTE If the View Status screen shows BT: No Signal even after you believe a BLE connection has been established, confirm that the remote station is actively transmitting TEL or CMD messages and not just maintaining a silent connection.

6.2 Message Protocol

The MockSat only receives messages. It does not transmit position data outward in the current firmware. All messages must be terminated with a newline character (line feed, 0x0A). Carriage return characters (0x0D) are silently discarded. The two supported message types are:

Message Type	Prefix	Format	Example	Effect
Telemetry	`TEL:`	`TEL:x,y,theta`	`TEL:1.5,3.2,90.0`	Updates telPos struct; triggers onTelemetryReceived()
Command	`CMD:`	`CMD:x,y,theta`	`CMD:5.0,2.1,45.0`	Updates cmdPos struct; triggers onCommandReceived()

x and y are floating-point position coordinates in your chosen unit system (meters, for example). theta is a floating-point heading angle in degrees. All three values must be present and separated by commas. A space is not permitted between the prefix and the values. Messages that do not match either prefix are logged to the Serial Monitor with a WARN tag and discarded.

Message Format Examples

```
// Valid message examples (newline-terminated):
TEL:0.0,0.0,0.0\n          -> Initial position, zero heading
TEL:1.500,3.200,90.000\n   -> Position (1.5, 3.2) heading 90 degrees
```

```
CMD:5.0,2.1,45.0\n          -> GTSS commands target position

// Invalid message examples (will be discarded with WARN):
tel:1.5,3.2,90.0\n          -> Lowercase prefix not recognized
TEL: 1.5,3.2,90.0\n         -> Space after colon causes parse failure
TEL:1.5,3.2\n               -> Missing theta field; sscanf returns 2, not 3
```

6.3 Bluetooth Menu: View Status

Navigate to Bluetooth > View Status to see the current connection state and BLE role.

LCD Line	Content	Meaning
Line 1	BT: Connected	A valid TEL or CMD message has been received from the remote station
Line 1	BT: No Signal	No valid data message received since boot or last reset
Line 2	Mode: Slave	HM-10 is in slave mode, advertising and waiting for incoming connections
Line 2	Mode: Master	HM-10 is in master mode, capable of scanning and initiating connections

6.4 Bluetooth Menu: Switch Mode

Navigate to Bluetooth > Switch Mode to toggle the HM-10 between slave and master roles. The current role and the action that K1 will take are displayed simultaneously:

LCD Line 1	LCD Line 2	Meaning
Current:	Slave K1=Master	Module is currently in slave mode. Press K1 to switch to master.
Current:	Master K1=Slave	Module is currently in master mode. Press K1 to switch to slave.

Pressing K1 sends the appropriate AT+ROLE command followed by AT+RESET to the HM-10. The module physically resets and applies the new role. A 1.5 second delay is used to allow the reset to complete before returning to the Bluetooth menu. The new role is reflected immediately in the View Status screen.

NOTE After switching roles and sending AT+RESET, the HM-10 module may take one to two seconds to become responsive. Any BLE connection that was active before the role switch will be dropped. The bluetoothConnected flag is not automatically cleared; it will reflect the previous state until a new message is received or a system reset is performed.

6.5 Bluetooth Menu: Scan Devices

Scanning for devices is only available when the HM-10 is in master mode. If you attempt to access Scan Devices while in slave mode, the LCD will display 'Master mode required first' for two seconds before returning to the Bluetooth menu.

When the HM-10 is in master mode and you select Scan Devices, the firmware sends AT+DISC? to the module and starts a 6-second scan timer. During scanning, the LCD displays:

LCD Line	Content	Meaning
Line 1	Scanning BLE...	Active scan in progress
Line 2	Found: N K1=Stop	N is the number of devices found so far. Press K1 to stop early.

The firmware parses HM-10 scan responses that begin with OK+DIS or OK+DISC:. It extracts the device identifier from the portion after the last colon. Up to 8 devices are stored. After the 6-second timeout, or when K1 is pressed, the system transitions to the Device List screen.

6.6 Bluetooth Device List and Connecting

After a scan, the device list screen shows one device at a time. Navigate with K2 (Up) and K3 (Down). The LCD shows the device index and total count on line 1, and the device MAC address or identifier on line 2 (truncated to 16 characters).

Press K1 to initiate a connection to the highlighted device. The firmware will strip colons from the MAC address and send AT+CON followed by the raw hex MAC string. A 2-second delay is used for the connection attempt, then the screen transitions to Bluetooth View Status to show the result.

NOTE If no devices are found, the screen shows 'No devices found' and K4 returns to the Bluetooth menu. Re-run a scan to try again. Ensure the remote device is powered, advertising, and within BLE range (typically up to 10 meters with line of sight).

6.7 HM-10 AT Command Reference

AT Command	Sent By Firmware	Response	Purpose
AT	Not sent automatically	OK	Test communication with the module
`AT+ROLE0`	btSetSlave()	OK+Set:0	Configure HM-10 as BLE slave (peripheral)
`AT+ROLE1`	btSetMaster()	OK+Set:1	Configure HM-10 as BLE master (central)
`AT+RESET`	After role change	OK+RESET	Reset the module to apply role change
`AT+DISC?`	btStartScan()	OK+DISC:... list	Discover nearby BLE devices (master mode only)
`AT+CONmac`	btConnectToDevice()	OK+CONN	Connect to device by raw MAC hex string

AT Command	Sent By Firmware	Response	Purpose
`AT+ADDR?`	Not sent automatically	OK+ADDR:mac	Query the module own MAC address
`AT+NAME?`	Not sent automatically	OK+NAME:name	Query the module device name

7. Pick Mode

The Pick Mode menu allows the operator to select a behavioral profile that determines how the eight motor channels are commanded. Selecting a mode executes the applyMode() function, which first stops all motors, then applies the mode-specific throttle assignments. The selected mode name is stored in the selectedMode variable and printed to the Serial Monitor. After selection, the display returns to the Main Menu.

Mode	Motor Behavior	Use Case
PointNav	All motors stopped. Placeholder for autonomous navigation logic.	Future integration point for a position PID controller using localPos and cmdPos
Xbox Control	All motors stopped. Placeholder for joystick input passthrough.	Future integration for an Xbox controller connected over USB or BLE
Idle	All motors stopped explicitly.	Safe state with no active thrust; useful during bench testing
Full Throttle	All 8 motors commanded to 100%.	Maximum thrust test; do not run unrestrained
Half Throttle	All 8 motors commanded to defaultThrottle %.	Configurable thrust test; defaultThrottle set in Settings menu
Spin +Z	Even-numbered motors (2,4,6,8) at defaultThrottle %.	Positive Z-axis rotation profile for attitude control testing
Spin -Z	Odd-numbered motors (1,3,5,7) at defaultThrottle %.	Negative Z-axis rotation profile for attitude control testing
Other	All motors stopped. Placeholder for custom modes.	Reserved for custom user-defined motor control profiles

WARNING Full Throttle commands all eight motors to 100% immediately. If physical ESCs and motors are connected, this will spin all thrusters at maximum power. Only use this mode in a secured test environment with the vehicle properly restrained.

NOTE The Spin +Z and Spin -Z modes command either even or odd motors based on array index, not on physical layout. The actual rotation direction depends entirely on the physical orientation and rotation direction of each motor on your specific vehicle.

8. Settings

The Settings menu provides access to three adjustable parameters and an IMU recalibration trigger. Settings adjustments are made on dedicated screens using K2 and K3, and confirmed by pressing K1. Pressing K4 from any adjustment screen discards all changes and returns to the Settings menu without saving.

8.1 Default Throttle

Default Throttle sets the percentage value used by the Half Throttle, Spin +Z, and Spin -Z operating modes. It is stored in the variable defaultThrottle.

Property	Value
Default value	50.0%
Adjustment step	5.0% per K2/K3 press
Minimum	0.0%
Maximum	100.0%
Saved to	defaultThrottle (volatile; resets on power loss)

8.2 Motor Test Duration

Motor Test Duration sets the number of seconds used in future timed motor test routines. It is stored in the variable motorTestDuration. There is no active motor test routine in the current firmware; this setting is a configuration parameter for future implementation.

Property	Value
Default value	3 seconds
Adjustment step	1 second per K2/K3 press
Minimum	1 second
Maximum	10 seconds
Saved to	motorTestDuration (volatile; resets on power loss)

8.3 IMU Update Rate

IMU Update Rate controls how frequently the firmware polls all BNO055 sensor vectors and refreshes the IMU data variables. A lower value means more frequent updates (higher CPU usage). A higher value reduces polling frequency but also reduces the freshness of data shown in diagnostics. This value is stored in both imuUpdateRate and SENSOR_UPDATE_INTERVAL.

Property	Value
Default value	100 ms (10 Hz)
Adjustment step	50 ms per K2/K3 press
Minimum	50 ms (20 Hz)
Maximum	500 ms (2 Hz)
K2 (Up) behavior	Decreases the interval (faster polling)
K3 (Down) behavior	Increases the interval (slower polling)
Saved to	imuUpdateRate and SENSOR_UPDATE_INTERVAL (volatile)

NOTE Setting the IMU rate to 50ms (20 Hz) is close to the BNO055 native sensor fusion output rate. Going faster than the sensor output rate will return repeated values and wastes CPU cycles without benefit.

8.4 Calibrate IMU

Selecting Calibrate IMU from the settings menu immediately launches the same CalibrateIMU() routine that runs during boot. The LCD will show the live calibration scores (S, G, M) and block until all three reach 3. When complete, the zero-reference orientation is updated and the system returns to the Settings menu.

Use this option any time the operating environment changes significantly (for example, after moving the system to a different location, or after the magnetometer has been exposed to strong magnetic fields) to re-establish a valid orientation baseline.

9. Diagnostics

The Diagnostics menu provides live readouts of all major sensor and system state variables. Data is updated at the IMU polling interval set in Settings. Navigate between the six diagnostic screens using K2 and K3 from the Diagnostics menu, then press K1 to enter. Press K4 to return to the Diagnostics menu from any screen.

9.1 IMU Data (5 Screens)

The IMU Data screen displays BNO055 sensor vector data. There are five sub-screens, cycled with K2 and K3. All values are read in the sensor frame and are offset-corrected for Euler angles based on the zero-reference captured during calibration.

Screen	LCD Line 1	LCD Line 2	Sensor Vector	Units
0 (Euler)	Euler X: value	Y: value Z: value	VECTOR_EULER	Degrees (offset-corrected)
1 (Gyro)	Gyro X: value	Y: value Z: value	VECTOR_GYROSCOPE	Rad/s
2 (Accel)	Accel X: value	Y: value Z: value	VECTOR_ACCELEROMETER	m/s^2 (total)
3 (Mag)	Mag X: value	Y: value Z: value	VECTOR_MAGNETOMETER	uT (microtesla)
4 (Linear Acc)	Lin X: value	Y: value Z: value	VECTOR_LINEARACCEL	m/s^2 (gravity-free)

NOTE Euler X is the heading (yaw) angle relative to the zero-reference. Euler Y is pitch. Euler Z is roll. All three are wrapped to the range -180 to +180 degrees by the firmware after applying the calibration offset.

9.2 Temperature

The Temperature screen displays the BNO055 internal chip temperature in degrees Celsius. This value is read using bno.getTemp() and represents the die temperature of the sensor, which may be several degrees warmer than ambient air temperature due to self-heating.

9.3 Battery

The Battery screen is a placeholder. It displays 'Battery Data' on line 1 and 'Not Available' on line 2. No battery monitoring circuit is implemented in the current firmware. This screen is reserved for a future analog voltage divider circuit connected to a Teensy ADC pin.

9.4 Motors

The Motors diagnostic screen displays the current throttle percentage for all eight motor channels simultaneously. Motor indices 1 through 4 appear on line 1, and 5 through 8 appear on line 2. Values are displayed as integers with no decimal places to fit within the 16-character LCD width.

```
Example Motors Display (Full Throttle on M3; Half Throttle on M6 and M8)
M1-4: 0  0  100  0
M5-8: 0  50  0  50
```

NOTE If the motorsArmed flag is false (which should never happen after a normal boot), all motor commands will be silently rejected and throttle values in this screen will remain at 0.

9.5 Position (EST vs TEL)

The Position screen shows two rows of position data enabling comparison between what the onboard navigation system believes the current position to be and what was last received over BLE telemetry.

Row	Prefix	Source	Updated By
Line 1	EST:	localPos struct (onboard estimate)	Must be written by a nav/control module (currently always 0.0,0.0,0.0)
Line 2	`TEL:`	telPos struct (BLE telemetry)	Updated every time a valid TEL: message is received over BLE

The display format is PREFIX:x,y,theta where x and y are shown to one decimal place and theta is shown as a whole number. All values are truncated to fit within 16 characters.

```
Example Position Display
EST:0.0,0.0,0      <- Local nav estimate
TEL:1.5,3.2,90     <- Last received BLE telemetry
```

9.6 Command (CMD vs TEL)

The Command screen shows the most recent commanded target position from the GTSS alongside the last received telemetry position. This screen is intended for operators to verify that the MockSat has received a valid command and to observe how far it currently is from the target.

Row	Prefix	Source	Updated By
Line 1	`CMD:`	cmdPos struct (GTSS command)	Updated every time a valid CMD: message is received over BLE
Line 2	`TEL:`	telPos struct (BLE telemetry)	Updated every time a valid TEL: message is received over BLE

```
Example Command Display
CMD:5.0,2.1,45     <- GTSS commanded target
TEL:1.5,3.2,90     <- Current telemetry position
```

NOTE Both CMD and TEL values initialize to 0.0, 0.0, 0.0 on boot and after a system reset. If no BLE messages have been received, this screen will show all zeros for both rows. This is normal and does not indicate a fault.

10. System Reset

The Reset option in the Main Menu provides a soft system reset. It is not a power cycle or hardware reset; it re-runs the boot sequence entirely in firmware, re-initializes all state variables, and re-performs IMU calibration.

10.1 Reset Confirmation

Selecting Reset from the Main Menu navigates to a confirmation screen:

```
Reset Confirmation LCD Display
Reset System?
K1=Yes  K4=No
```

Press K1 to confirm and execute the reset. Press K4 to cancel and return to the Main Menu.

10.2 What a Reset Clears

Variable / State	Reset Value	Notes
All motor throttles	0.0 for all 8	stopAllMotors() is called before reset
bluetoothConnected	false	Connection status cleared; remote must resend data
telPos	0,0,0	Last received telemetry position cleared
cmdPos	0,0,0	Last received command position cleared
localPos	0,0,0	Local nav estimate cleared
mainMenuCursor	0	Main menu returns to top item (Pick Mode)
selectedMode	None	Operating mode selection cleared
bootStep	0	Boot sequence restarts from step 0
IMU calibration	Re-performed	Full calibration procedure runs again during boot
imuUpdateRate	Unchanged	Settings values are NOT cleared by reset
defaultThrottle	Unchanged	Settings values are NOT cleared by reset
motorTestDuration	Unchanged	Settings values are NOT cleared by reset

WARNING Settings values (defaultThrottle, motorTestDuration, imuUpdateRate) are stored in volatile RAM and are NOT reset by the soft reset. They only revert to their defaults (50%, 3s, 100ms) if the device is completely power-cycled.

11. Serial Monitor Reference

All significant firmware events are logged to the USB Serial port at 115200 baud. The Serial Monitor is an essential tool for debugging Bluetooth message parsing, confirming button inputs, tracking state transitions, and monitoring motor commands.

Prefix Tag	Source Function	Example Output	Meaning
[BOOT]	handleBootSequence()	[BOOT] IMU BNO055: OK	Boot sequence step completed
[INPUT]	handleButtons()	[INPUT] K1 - Select	Button press detected
[BT] <<<	handleBluetooth()	[BT] <<< Raw: TEL:1.5,3.2,90.0	Raw line received from HM-10
[BT] >>>	sendATCommand()	[BT] >>> AT: AT+ROLE1	AT command sent to HM-10
[BT] AT	handleATResponse()	[BT] AT Response: OK+DISC:1	AT response received and parsed
[BT] Found	handleATResponse()	[BT] Found device: 7C4B46XXXXXX	BLE device discovered during scan
[TEL]	onTelemetryReceived()	[TEL] Received position -> x: 1.50 y: 3.20	Parsed TEL message content
[CMD]	onCommandReceived()	[CMD] GTSS target -> x: 5.00 y: 2.10	Parsed CMD message content
[MODE]	handleSelect()	[MODE] Selected: Full Throttle	Operating mode changed
[MOTOR]	setMotorThrottle()	[MOTOR] M3 -> 100%	Individual motor throttle command
[MOTOR]	stopAllMotors()	[MOTOR] Stop	All motors commanded to zero
[SETTINGS]	handleSelect()	[SETTINGS] Throttle: 75%	Setting value confirmed and saved
[SYSTEM]	handleSelect()	[SYSTEM] Reset.	Soft reset executed
[WARN]	parseMessage()	[WARN] Malformed TEL message.	Malformed or unknown BLE

Prefix Tag	Source Function	Example Output	Meaning
			message received
[IMU]	handleDown()	[IMU] Screen: 2	IMU diagnostics sub-screen changed

11.1 Calibration Output Example

```
IMU Calibration Serial Output
Calibration Complete
Zero ref - X:182.44 Y:-1.22 Z:0.58
```

11.2 Complete BLE Message Exchange Example

```
Serial Monitor: Complete BLE Session Example
[BT] <<< Raw: TEL:1.5,3.2,90.0
[BT] Connection detected - first data message received.
[TEL] Received position -> x: 1.50  y: 3.20  theta: 90.00

[BT] <<< Raw: CMD:5.0,2.1,45.0
[CMD] GTSS target -> x: 5.00  y: 2.10  theta: 45.00

[BT] <<< Raw: TEL:1.8,3.4,88.0
[TEL] Received position -> x: 1.80  y: 3.40  theta: 88.00
```

12. Developer Extension Points

The firmware is designed with several clearly marked extension points where integrators can add custom logic. Each is identified by a TODO comment in the source code. The following table documents these points and provides guidance on how to integrate with them.

Location	TODO Comment	Integration Guidance
setMotorThrottle()	PWM output via HiLetgo PWM board	Instantiate an Adafruit_PWMServoDriver object, call setPWM() with the mapped motor channel and the computed microsecond pulse width for your ESC.
onCommandReceived()	Pass cmdPos + localPos to motion controller	Implement a PID or other control law using cmdPos as the setpoint and localPos as the process variable. Output to setMotorThrottle() calls.
applyMode() case 0	PointNav stub	Implement the autonomous point navigation entry logic here. Typically sets a target in cmdPos and enables the motion controller.
applyMode() case 1	Xbox Controller stub	Initialize a USB HID host or BLE gamepad connection and begin reading joystick axes to drive motor throttle commands.
localPos variable	Updated from nav controller	Write your estimated position (from odometry, IMU integration, or external positioning) to localPos.x, localPos.y, and localPos.theta to populate the Position diagnostics screen.

12.1 Adding a Custom Operating Mode

To add a new operating mode beyond the existing eight, expand the modes array in the PICK_MODE_MENU case of handleSelect(), update the pickModeCursor wrap value from 7 to the new maximum, add a corresponding String in the PICK_MODE_MENU case of updateDisplay(), and add a new case to the applyMode() switch statement.

Adding a Custom Mode: Code Pattern

```
// Step 1: In handleSelect() PICK_MODE_MENU case:
String modes[] = { ..., "MyMode" };  // Add new name

// Step 2: In handleUp() and handleDown() for PICK_MODE_MENU:
if (pickModeCursor > 8) pickModeCursor = 0;  // Extend wrap value
if (pickModeCursor < 0) pickModeCursor = 8;

// Step 3: In updateDisplay() PICK_MODE_MENU case:
String modes[] = { ..., "MyMode" };  // Mirror the same array

// Step 4: In applyMode():
case 8: myModeFunction(); break;
```

12.2 Adding a Telemetry Transmit Channel

The current firmware is receive-only. To transmit localPos or IMU data back to the GTSS, add BLE.println() calls in the main loop or in a dedicated transmit function. The HM-10 in slave mode will forward any data written to Serial1 over BLE to the connected master device. Use the same message format for consistency:

Example: Transmitting Local Position Over BLE

```
// Example: Transmit local position at 1 Hz
if (millis() - lastTxTime >= 1000) {
  String msg = "LOC:" + String(localPos.x, 2) + ","
               + String(localPos.y, 2) + ","
               + String(localPos.theta, 1);
  BLE.println(msg);
  lastTxTime = millis();
}
```

13. Troubleshooting

Symptom	Likely Cause	Resolution
LCD remains blank after power-up	Wrong I2C address or no power to LCD backpack	Scan I2C bus with an Arduino I2C scanner sketch. Default address is 0x27. Try 0x3F if not found. Verify 5V or 3.3V on VCC.
Boot stuck at Calibrating IMU indefinitely	IMU not reaching S:3 G:3 M:3	Perform the full calibration motion: hold still (gyro), figure-eight (mag), six-face rest (accel). Move away from magnetic interference sources.
IMU shows Status: N/A at boot	BNO055 not detected on I2C at 0x28	Check SDA/SCL connections to pins 18/19. Verify I2C address. Check 3.3V on VCC. Confirm ADR pin is tied to GND.
BT: No Signal even after connecting	Remote device has not sent a TEL or CMD message	Confirm the GTSS is actively transmitting well-formed TEL: or CMD: messages terminated with newline (0x0A).
TEL or CMD values always 0.0,0.0,0	No messages received yet or messages malformed	Open Serial Monitor, check for [WARN] tags. Verify message format exactly matches TEL:x,y,theta with no spaces.
Scan Devices finds 0 devices	HM-10 not in master mode, or no devices nearby	Navigate to BT > Switch Mode and confirm Mode: Master is shown. Ensure remote device is powered and advertising.
`AT+CON fails to connect`	MAC address format issue	HM-10 expects a 12-character hex MAC with no colons. The firmware strips colons automatically. Check Serial Monitor for the exact AT+CON string being sent.
Buttons not responding	Wiring error or wrong INPUT_PULLUP logic	Confirm button is wired between the signal pin and GND. With INPUT_PULLUP, pin reads HIGH when open and LOW when pressed. Check for short circuits.
All motor values stuck at 0	motorsArmed is false	This should not occur after a normal boot. Check Serial Monitor for [MOTOR] Arming ESCs. If not present, check for early boot failure.
Settings reset after power cycle	Settings stored in volatile RAM	This is expected behavior. Settings are not stored in EEPROM or flash. Implement EEPROM.write() calls in the save handlers to add persistence.

14. Quick Reference Card

14.1 Button Cheat Sheet

Context	K1 (Select)	K2 (Up)	K3 (Down)	K4 (Back)
Any list menu	Enter item	Cursor up (wraps)	Cursor down (wraps)	Return to parent
Settings adjust	Save and return	Increase value	Decrease value	Cancel, discard
IMU Diagnostics	N/A	Previous screen	Next screen	Return to menu
BT Mode screen	Toggle role + reset	N/A	N/A	Return to BT menu
BT Scan	Stop scan early	N/A	N/A	Stop scan + back
BT Device List	Connect to device	Previous device	Next device	Return to BT menu
Confirm Reset	Execute reset	N/A	N/A	Cancel

14.2 BLE Message Format Summary

Message	Format	Example	Result
Telemetry	`TEL:x,y,theta\n`	`TEL:1.5,3.2,90.0\n`	Updates telPos; sets bluetoothConnected = true
Command	`CMD:x,y,theta\n`	`CMD:5.0,2.1,45.0\n`	Updates cmdPos; sets bluetoothConnected = true

14.3 IMU Calibration Score Reference

Score	Meaning
0	Not calibrated; sensor readings may be unreliable
1	Partially calibrated
2	Mostly calibrated; usable for non-critical applications
3	Fully calibrated; boot will proceed past this sensor

14.4 Settings Limits Reference

Setting	Default	Min	Max	Step
Default Throttle	50%	0%	100%	5%
Motor Test Duration	3s	1s	10s	1s
IMU Update Rate	100ms	50ms	500ms	50ms