November 2025

# Cygnus Reach Device Overview

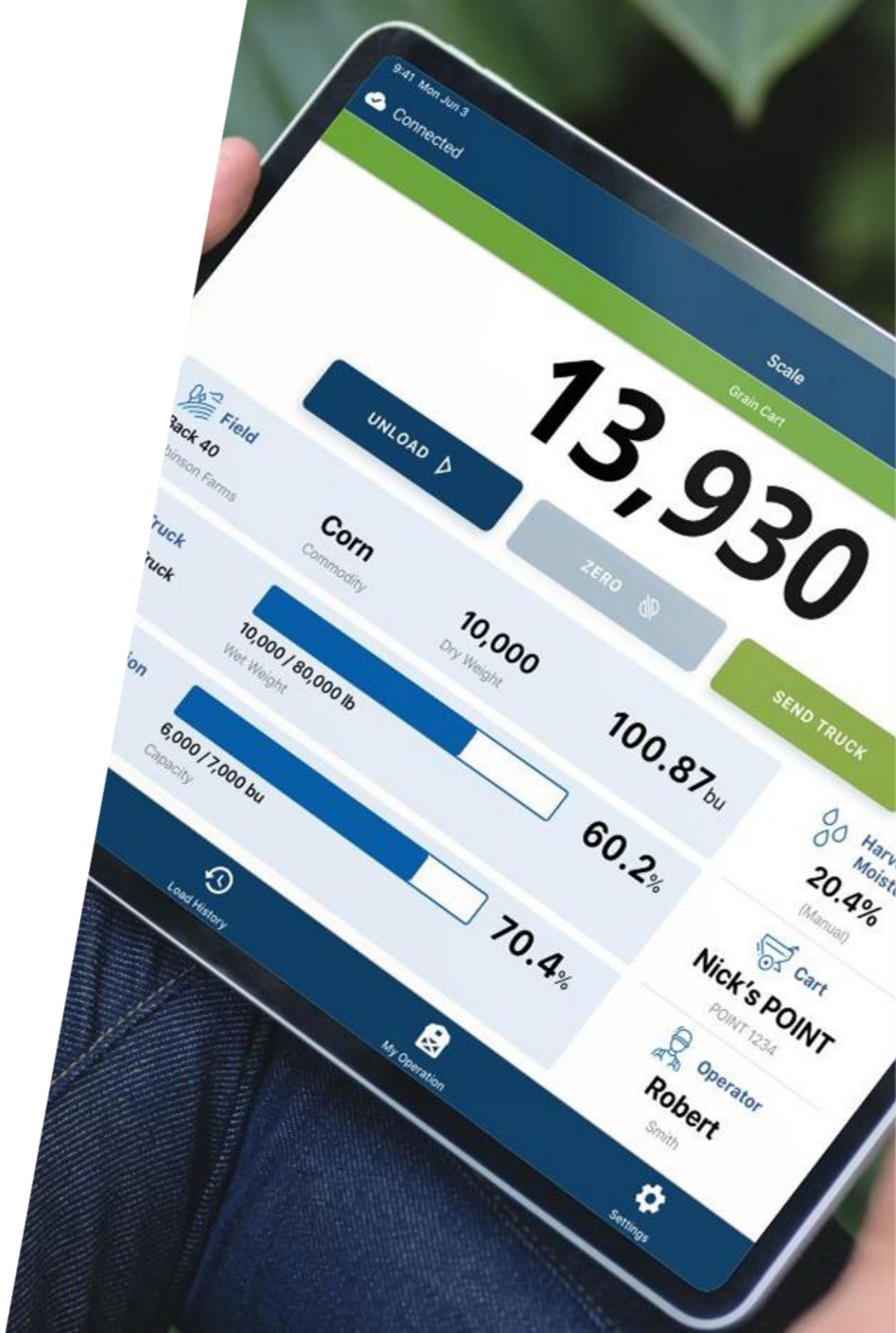
What Reach Brings to an Embedded Device

# OVERVIEW

01

# Cygnus Reach

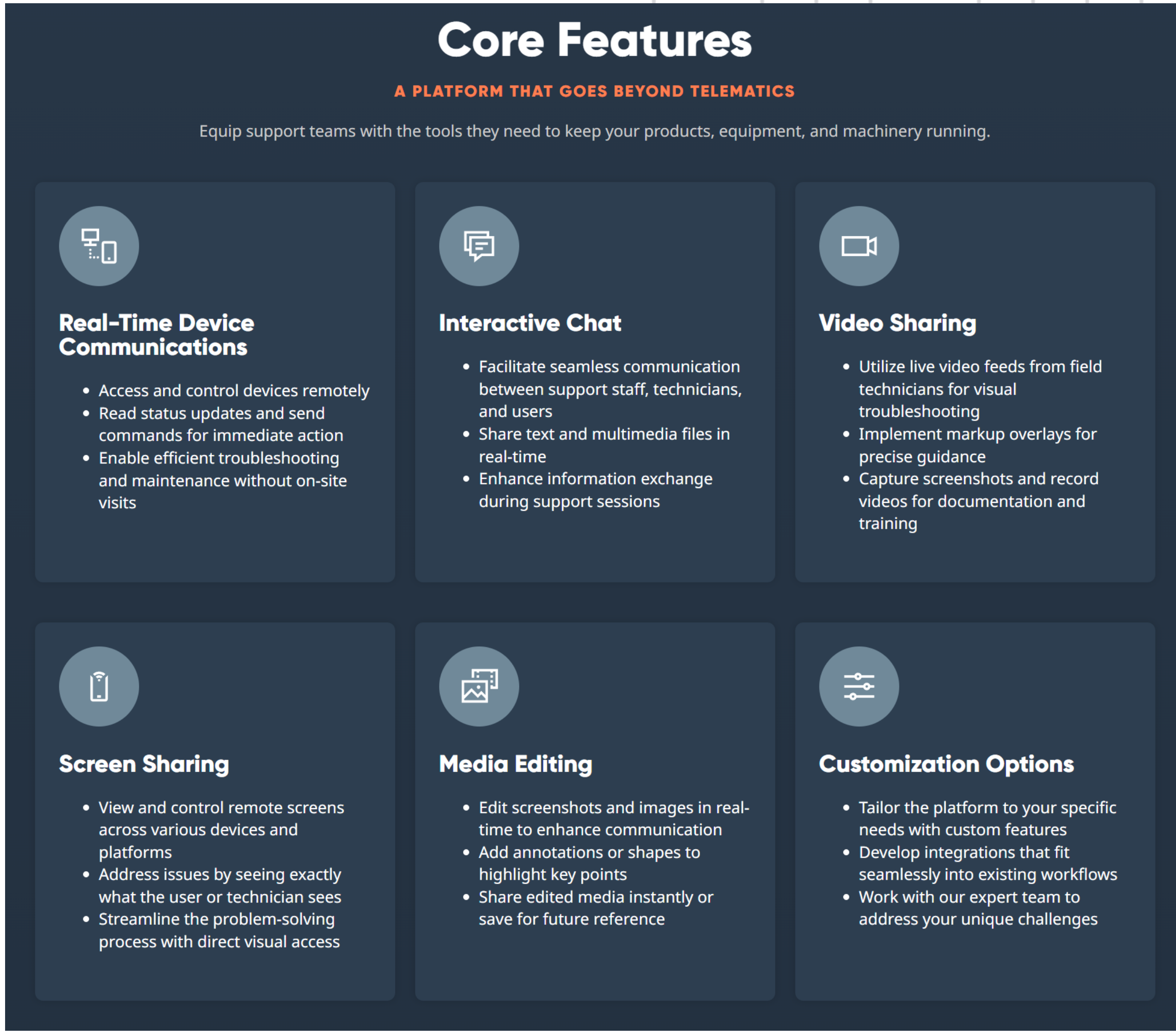
What is Cygnus, vs What is Reach?



# Cygnus

## Reach

- Cygnus is the name of a brand of i3, owned by Helios
- Reach is the remote support system that includes a cloud backend to support a data pipeline, video link to a customer's phone as well as phone screen sharing.
- The Reach protocol is used by Reach to expose the internals of a device.
    - Basic Reach support can be implemented without using the Reach protocol, though at increased development in the app.
- Integrating the Reach protocol in your device gives greater depth to support features



## Core Features

**A PLATFORM THAT GOES BEYOND TELEMATICS**

Equip support teams with the tools they need to keep your products, equipment, and machinery running.

### Real-Time Device Communications

- Access and control devices remotely
- Read status updates and send commands for immediate action
- Enable efficient troubleshooting and maintenance without on-site visits

### Interactive Chat

- Facilitate seamless communication between support staff, technicians, and users
- Share text and multimedia files in real-time
- Enhance information exchange during support sessions

### Video Sharing

- Utilize live video feeds from field technicians for visual troubleshooting
- Implement markup overlays for precise guidance
- Capture screenshots and record videos for documentation and training

### Screen Sharing

- View and control remote screens across various devices and platforms
- Address issues by seeing exactly what the user or technician sees
- Streamline the problem-solving process with direct visual access

### Media Editing

- Edit screenshots and images in real-time to enhance communication
- Add annotations or shapes to highlight key points
- Share edited media instantly or save for future reference

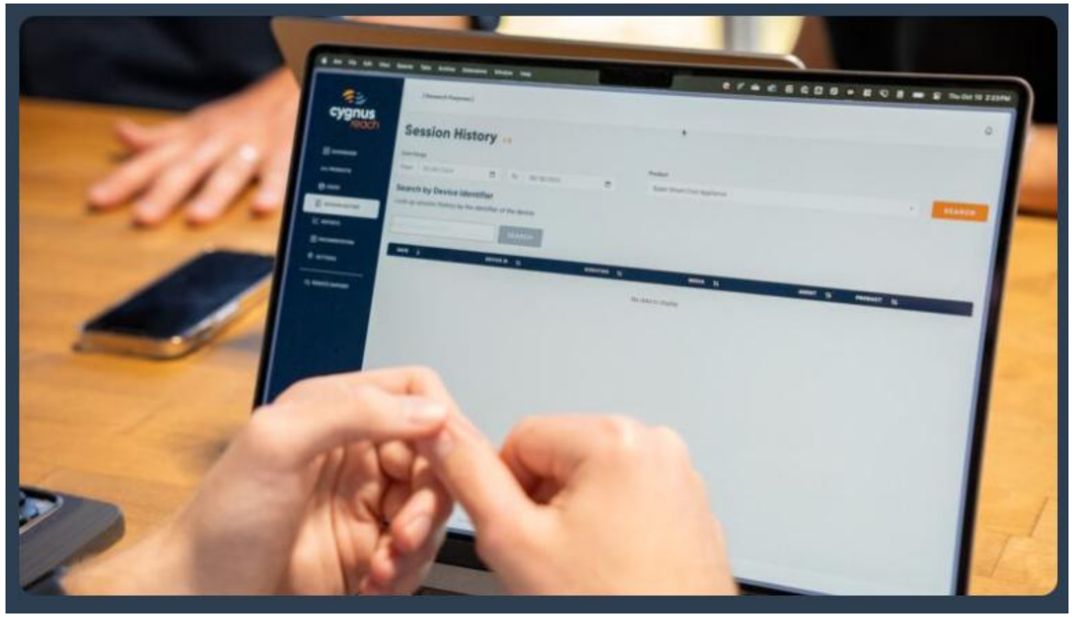
### Customization Options

- Tailor the platform to your specific needs with custom features
- Develop integrations that fit seamlessly into existing workflows
- Work with our expert team to address your unique challenges

# Web Portal

## For the Support Agent

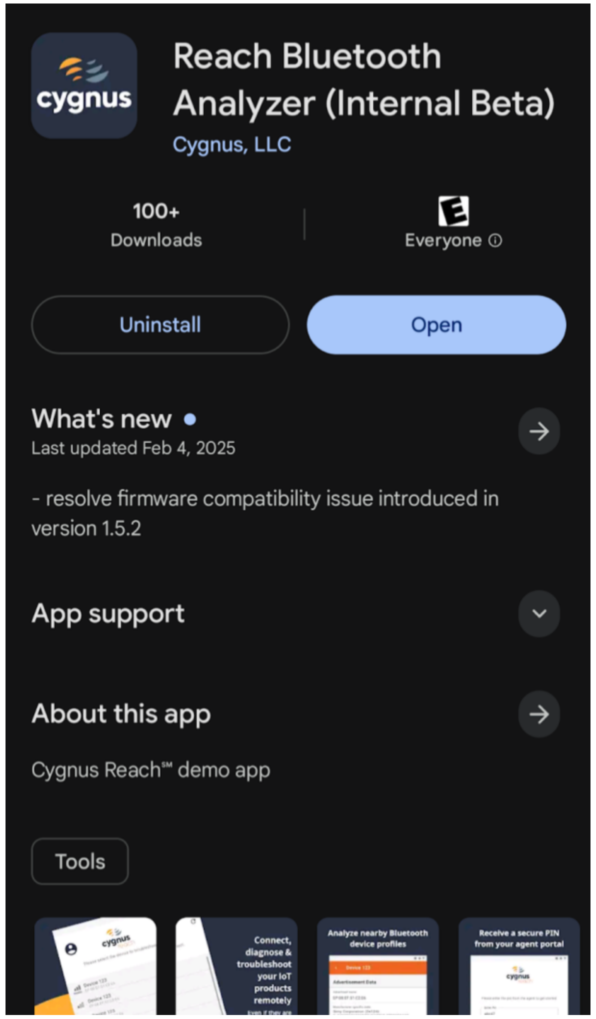
- Tech services uses the web portal to initiate support sessions
  - These are typically internal tech support people at the OEM/ODM
- The web portal allows the support agent to interact with the device as if they were next to it in BLE range, with the assistance of the end user on site
- Allows the support agent to view the app screen as well as see video from the phone that on site person.



# Apps

## For the onsite person

- An app is required to facilitate the remote support session
- Reach platform can be an independent app or built into an existing app
- BLE is used to communicate from the app to the device
- The app is typically used by the end user on site
  - Reach connect platform under development is for always connected products
- i3 provides “generic” Android and iOS apps that can be used in development.
- The Reach App can serve as a gateway to a web interface.
- The Reach App is typically customized for products.
  - Skin
  - Security



Reach Bluetooth Analyzer (Internal Beta)
Cygnus, LLC

100+ Downloads | Everyone

Uninstall | Open

What's new
Last updated Feb 4, 2025

- resolve firmware compatibility issue introduced in version 1.5.2

App support

About this app

Cygnus Reach℠ demo app

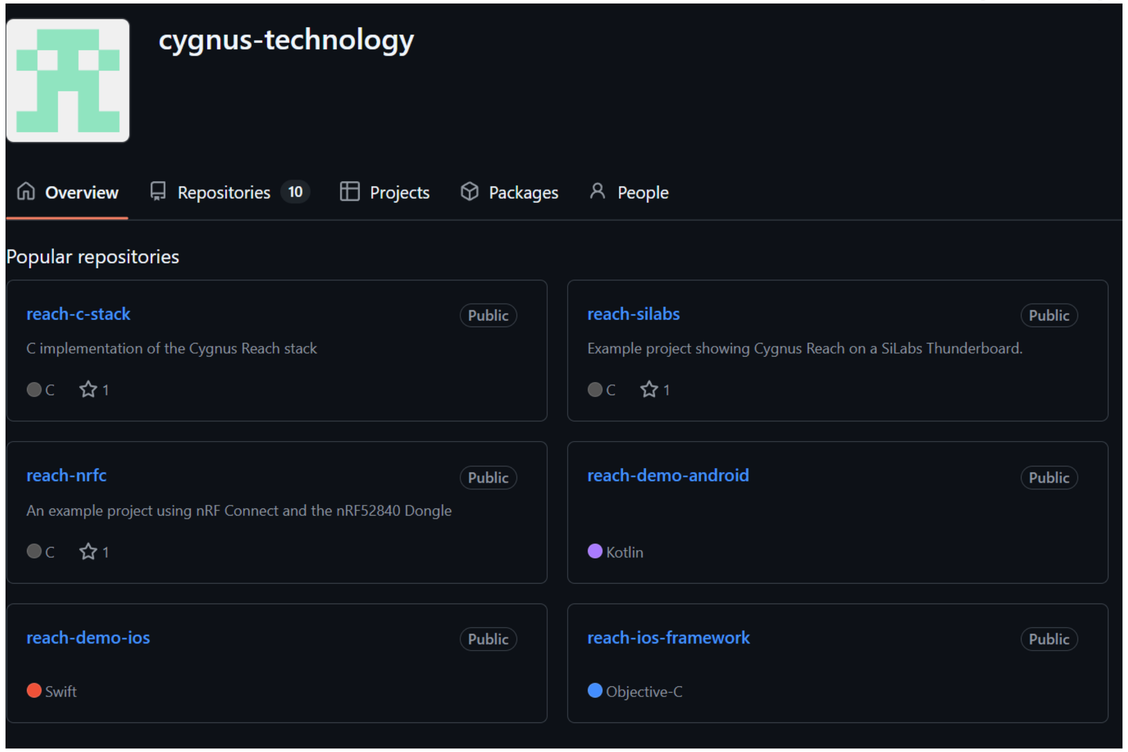
Tools

cygnus
Connect, diagnose & troubleshoot your IoT products remotely
Analyze nearby Bluetooth device profiles
Receive a secure PIN from your agent portal

## Reach

### Construction

- C code for the device (server)
  - Matching Java, C#, etc. on the app as a client
- Protobuf describes the serial communication stream.
  - Format compatible with OpenPV
- Open-source Git repositories stored on github
  - MIT license
  - https://github.com/cygnus-technology
- A defined and isolated "integration" connects to a BLE stack.
- A JSON script defines the capabilities of a specific device
- Reach Protocol communicates over BLE, these types of communicated are grouped into Services



Popular repositories

reach-c-stack Public
C implementation of the Cygnus Reach stack
C ☆ 1

reach-silabs Public
Example project showing Cygnus Reach on a SiLabs Thunderboard.
C ☆ 1

reach-nrfc Public
An example project using nRF Connect and the nRF52840 Dongle
C ☆ 1

reach-demo-android Public
Kotlin

reach-demo-ios Public
Swift

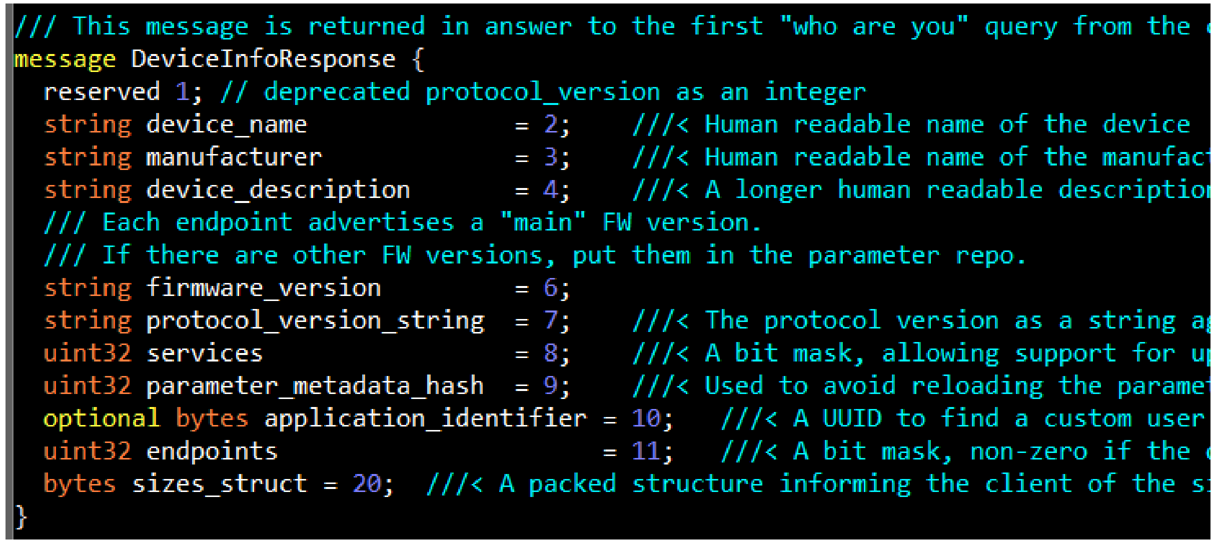
reach-ios-framework Public
Objective-C

# Reach Protocol

## Services

- Access to a Parameter Repository
- Issue “commands”
- Send and receive “files” (larger blocks of data)
- Access the device command line
- Set the RTC clock of the device
- Configure WiFi
- OTA

- Each of these services can easily be excluded and otherwise configured.
- Each of these services is supported by the “generic” reach app.



```
/// This message is returned in answer to the first "who are you" query from the
message DeviceInfoResponse {
  reserved 1; // deprecated protocol_version as an integer
  string device_name              = 2;    ///< Human readable name of the device
  string manufacturer             = 3;    ///< Human readable name of the manufac
  string device_description       = 4;    ///< A longer human readable descriptio
  /// Each endpoint advertises a "main" FW version.
  /// If there are other FW versions, put them in the parameter repo.
  string firmware_version         = 6;
  string protocol_version_string  = 7;    ///< The protocol version as a string a
  uint32 services                 = 8;    ///< A bit mask, allowing support for u
  uint32 parameter_metadata_hash  = 9;    ///< Used to avoid reloading the parame
  optional bytes application_identifier = 10;   ///< A UUID to find a custom user
  uint32 endpoints                      = 11;   ///< A bit mask, non-zero if the
  bytes sizes_struct = 20;  ///< A packed structure informing the client of the s
}
```

# Reach Protocol

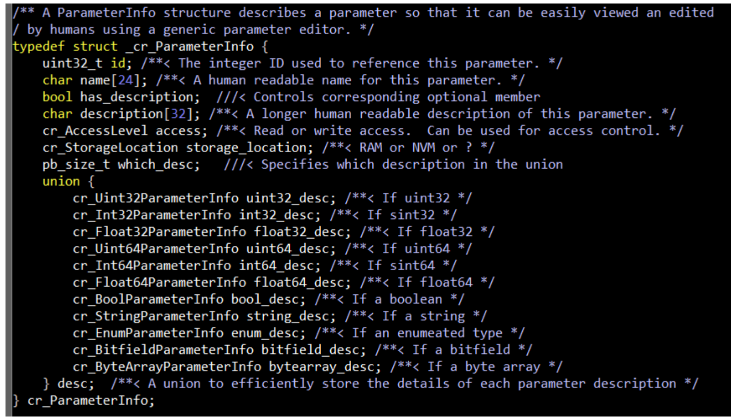
## OTA DFU

- "Over The Air Download Firmware Update"
  - Typically a feature provided with a BLE stack.
- The provider of the stack (SiLabs, Microchip) typically provides their own tool (app) for this purpose.
- The published protocol can be integrated into Reach apps.

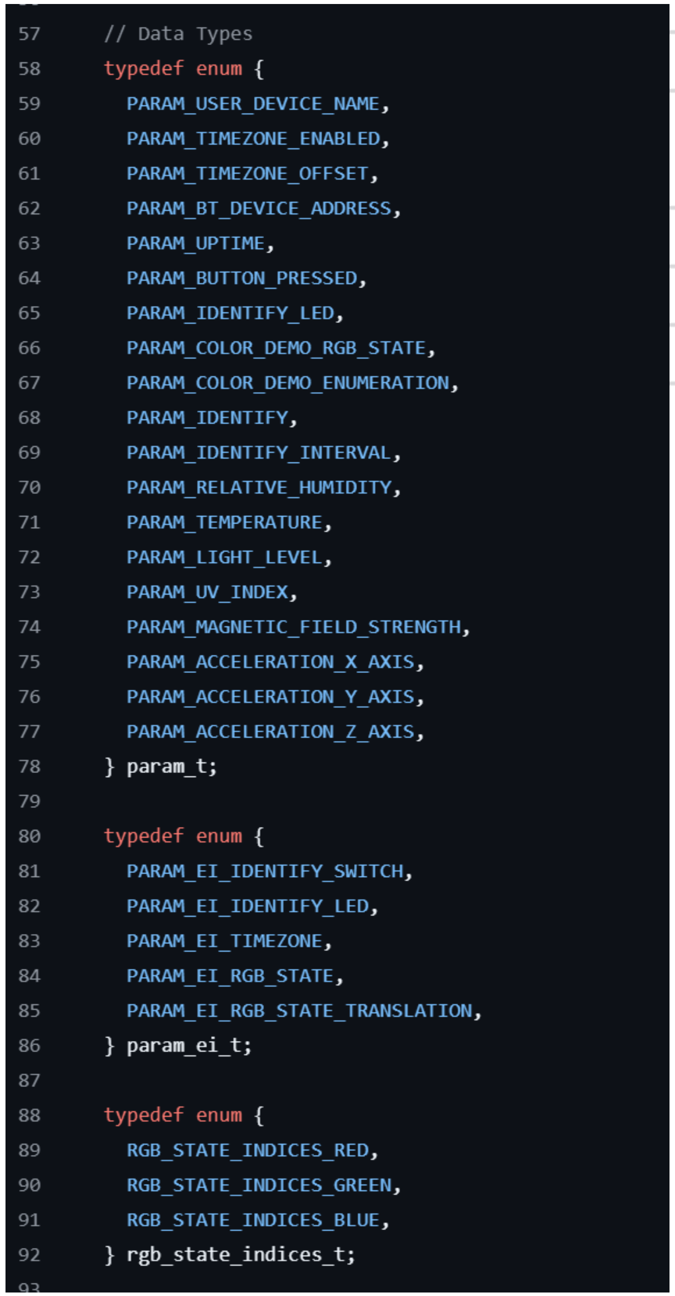
# Reach Protocol

## The Parameter Service

- Remote access to variables stored in a “parameter repository.”
- Varying types from ints and floats to 32-character strings.
- Each entry has a human readable name and description.
- Caching is used to minimize access to a long list of parameters.
- By default, each parameter requires 180 bytes
    - This can be reduced by limiting string sizes.
- Verbose descriptions are designed to describe the data to humans.



```
/** A ParameterInfo structure describes a parameter so that it can be easily viewed an edited
/ by humans using a generic parameter editor. */
typedef struct _cr_ParameterInfo {
    uint32_t id; /**< The integer ID used to reference this parameter. */
    char name[24]; /**< A human readable name for this parameter. */
    bool has_description;  ///< Controls corresponding optional member
    char description[32]; /**< A longer human readable description of this parameter. */
    cr_AccessLevel access; /**< Read or write access.  Can be used for access control. */
    cr_StorageLocation storage_location; /**< RAM or NVM or ? */
    pb_size_t which_desc;   ///< Specifies which description in the union
    union {
        cr_Uint32ParameterInfo uint32_desc; /**< If uint32 */
        cr_Int32ParameterInfo int32_desc; /**< If sint32 */
        cr_Float32ParameterInfo float32_desc; /**< If float32 */
        cr_Uint64ParameterInfo uint64_desc; /**< If uint64 */
        cr_Int64ParameterInfo int64_desc; /**< If sint64 */
        cr_Float64ParameterInfo float64_desc; /**< If float64 */
        cr_BoolParameterInfo bool_desc; /**< If a boolean */
        cr_StringParameterInfo string_desc; /**< If a string */
        cr_EnumParameterInfo enum_desc; /**< If an enumeated type */
        cr_BitfieldParameterInfo bitfield_desc; /**< If a bitfield */
        cr_ByteArrayParameterInfo bytearray_desc; /**< If a byte array */
    } desc;  /**< A union to efficiently store the details of each parameter description */
} cr_ParameterInfo;
```



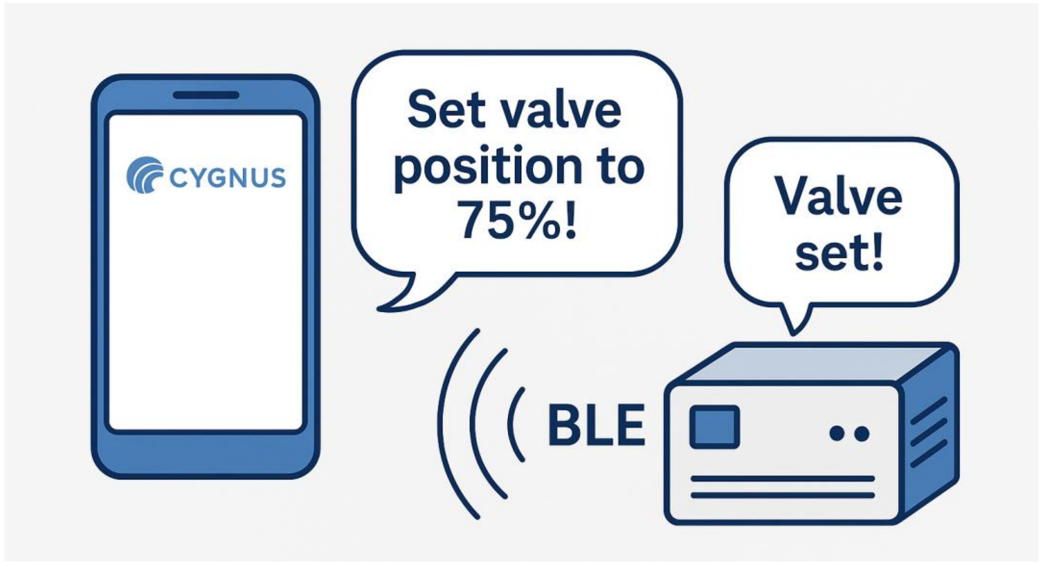
```
// Data Types
typedef enum {
  PARAM_USER_DEVICE_NAME,
  PARAM_TIMEZONE_ENABLED,
  PARAM_TIMEZONE_OFFSET,
  PARAM_BT_DEVICE_ADDRESS,
  PARAM_UPTIME,
  PARAM_BUTTON_PRESSED,
  PARAM_IDENTIFY_LED,
  PARAM_COLOR_DEMO_RGB_STATE,
  PARAM_COLOR_DEMO_ENUMERATION,
  PARAM_IDENTIFY,
  PARAM_IDENTIFY_INTERVAL,
  PARAM_RELATIVE_HUMIDITY,
  PARAM_TEMPERATURE,
  PARAM_LIGHT_LEVEL,
  PARAM_UV_INDEX,
  PARAM_MAGNETIC_FIELD_STRENGTH,
  PARAM_ACCELERATION_X_AXIS,
  PARAM_ACCELERATION_Y_AXIS,
  PARAM_ACCELERATION_Z_AXIS,
} param_t;

typedef enum {
  PARAM_EI_IDENTIFY_SWITCH,
  PARAM_EI_IDENTIFY_LED,
  PARAM_EI_TIMEZONE,
  PARAM_EI_RGB_STATE,
  PARAM_EI_RGB_STATE_TRANSLATION,
} param_ei_t;

typedef enum {
  RGB_STATE_INDICES_RED,
  RGB_STATE_INDICES_GREEN,
  RGB_STATE_INDICES_BLUE,
} rgb_state_indices_t;
```

# Reach Protocol

## The Command Service

- Devices often support a few commands.
    - Reboot
    - Factory Reset
- Reach gives a way to list and access such commands.
- Commands have a name, and ID and a description
- Commands execute a function on the device.

    - (per the picture) A better balloon might say "Set valve position according to all parameters"



# Reach Protocol

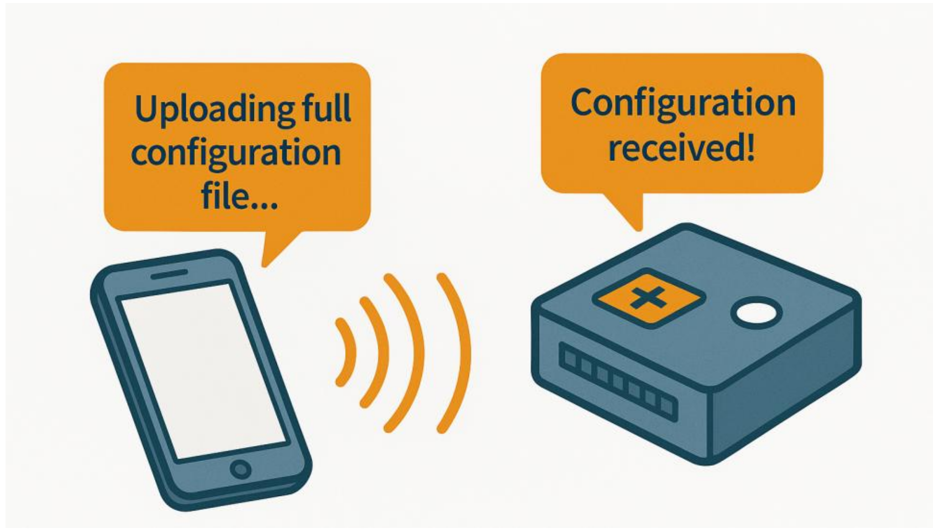
## Remote Command Line Interface

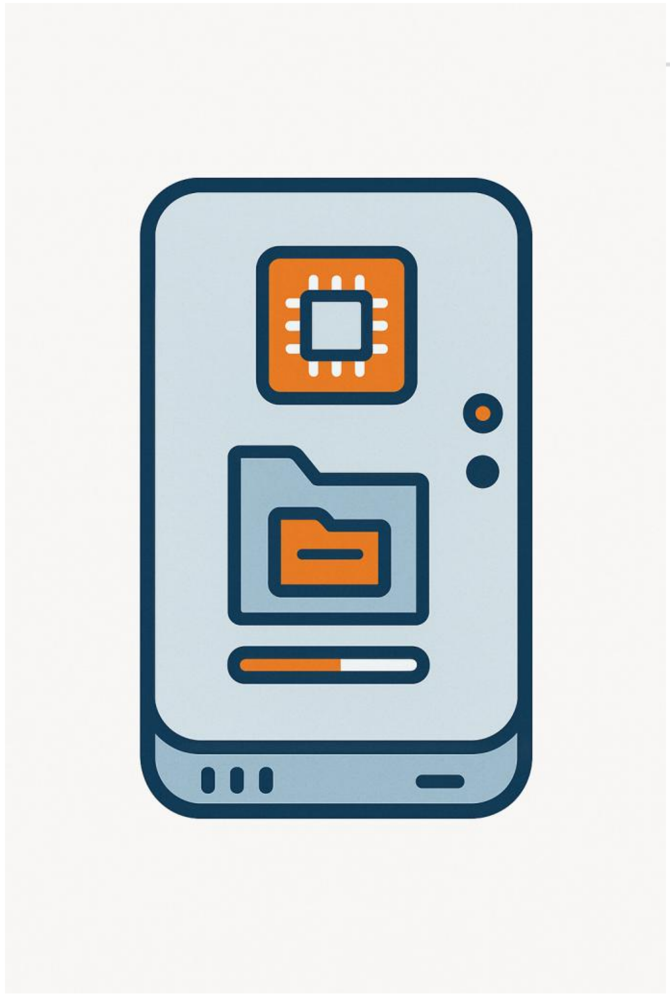
- Reach support remote access to the device command line.
- Allows a developer or support agent access to the debug and status features that might not have been included in the formal implementation of the Reach protocol.

# Reach Protocol

## The File Service

- Reach handles larger data structures with the “file” service.
- Any file system is device or application dependent. None is required.
- Files might be used to upload a complete configuration.
- Files might be used to by the support team to download a log.
- The file service provides a way to move a larger block of data.
- It says nothing about how the device manages this data.





## Reach Protocol

### Helpers

- Time Service
  - Small IoT devices often have a hardware RTC but no way to set it. This service is designed to make it easy to check and set an RTC.
- WiFi Service
  - Many IoT devices are able to access WiFi
  - Devices with no UI can be hard to configure for WiFi
  - The WiFi service allows a user to set the WiFi SSID and PW via the BLE interface to the phone.
  - Excluded when the device has no WiFi.

This page intentionally left blank.

# Reach Protocol

## Security

- The BLE link can be configured with the standard BLE security features.
- Support accounts can have different levels of access to features:
    - User may have the most limited access
    - FAE may have more access
    - Factory support may have full access

# Reach Code Architecture

Come Git it

- Git Submodules
  - reach-protobuf
    - Shared with and used by all Cygnus clients (apps, web, etc)
    - Defines communication protocol
  - reach-c-stack
    - C code implementing the stack
    - Shared with and used by all Cygnus embedded devices (servers)
  - reach-util
    - Python code generating tool
    - Reads JSON, writes C code
- Submodules are used as these codes are shared.
  - Have you used git submodules?

## Reach Code Architecture

### Local Reach Components

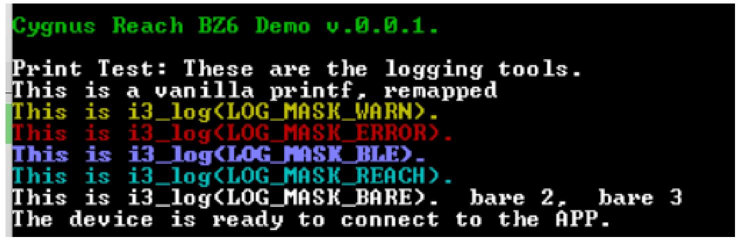
- Integrations/BZ6
  - Adapts Reach to the local BLE system
  - Should be shared by all BZ6 projects
- reach_bz6
  - Application Specific
  - Generated by CodeGenerator.py
    - Operating on reach_bz6.json
  - User code is added between comment blocks

## Other Services

- Commands are cheap.
  - Most products do not have very many commands.
- The time service is similarly cheap.
  - An RTC is a good idea especially to support long term logging.
- The file service can give people a way to upload (or download) a configuration file.
- The file service can support the retrieval of a relatively long duration log.
  - We probably have enough memory.

# Reach Logging Features

- The Reach package provides logging and error handling tailored for embedded environments
- i3_assert()
  - i3_assert() is designed to catch “programming errors.” As the programmer, you are saying “the code should never get here, so tell me right away if it does.”
  - This is as opposed to application errors that can be handled.
  - I3_assert() halts, printing where it is, and stopping at a breakpoint in the debugger.
- I3_log()
  - Like printf(), goes to the console.
  - Has an enable flag as the first parameter to support run time enable or disable of logs on a “per feature” basis.
  - It’s a 32 bit flag. Reach uses 8 of these bits. The app can use the rest.
  - Supports color coding using ANSI color codes



```
Cygnus Reach BZ6 Demo v.0.0.1.

Print Test: These are the logging tools.
This is a vanilla printf, remapped
This is i3_log(LOG_MASK_WARN).
This is i3_log(LOG_MASK_ERROR).
This is i3_log(LOG_MASK_BLE).
This is i3_log(LOG_MASK_REACH).
This is i3_log(LOG_MASK_BARE).  bare 2,  bare 3
The device is ready to connect to the APP.
```

# BZ6 App Architecture

## Using FreeRTOS

### One OS to rule them all

- Everything of the “app” happens in a task.
- BLE is supported on the BZ6 with a task.
  - Sources are not provided.
- BLE communicates with “the app” via an RTOS queue.
  - The queue is polled in the demos
  - A dedicated task could wake up on it.
- Can our applications use a wrapper to support a possible migration to a different RTOS?
  - Microchip provides osal_freertos.h which has most of the common things.
    - It's missing sleep, but we could extend it.
  - The CMSIS OSAL is attractive, but Microchip doesn't directly support it.
- We should ensure that the priority of all tasks defined in one place (task_priorities.h)
  - Here also briefly document the purpose of each task.
    - Eg: blocks on X, polls every N ms for Y, handles peripheral Z.

# Thread Safe Programming

## Lets unravel this one

- Access to the parameter repositories will probably have to be thread safe.
- Console logging will have to be thread safe.
- In a device like this that has threads, the features and the thread population will grow.
  - We must take care to define our thread safety rules from the start.
  - Clearly documenting the purpose of each thread and its communication patterns is a big help.
  - Global variables have their place, but be careful.
  - Defining variables as static forces another programmer to think twice before accessing it from another task.
- Minimize code in interrupts (signal a thread)

## Malloc and Free

- Prefer static allocation for memory that is always used.
- The examples use malloc and free:
  - OSAL_Malloc(), OSAL_Free()
- Use sparingly in your app:
- Use where the use is temporary.
- Carefully verify the corresponding free.
- We must install tracing, reporting and checking.
  - Clearly understand where is the heap, what is its state.
    - FreeRTOS has good features to support this.
    - Watch out for use of malloc by (BLE) libraries, as these are not traced.

| *Feature* | *Purpose* |
|---|---|
| xPortGetFreeHeapSize() | Current free heap |
| xPortGetMinimumEverFreeHeapSize() | Historical low-water mark |
| vApplicationMallocFailedHook() | Catch malloc failures |
| traceMALLOC() / traceFREE() | Instrument allocations/frees |