29 March 2018

Telstra IoT CAT-M1 Development Kit

Arduino Library Reference

Project	**TEL-IOT5 Telstra CAT-M1 Development Kit 2018**
Client	**Telstra**
Prepared by	**S. Khanna, K. Riegel,**
Revision	**R02**

TelstraM1Interface Class

Overview

The TelstraM1Interface class provides low-level functions for sending data and commands to the cellular system and receiving responses. This class is used extensively by higher-level classes such as TelstraM1Device and Connection4G. Its member functions are not intended to be called directly from Arduino sketches.

Constructor

TelstraM1Interface ()

Parameters
None.

Return Value
None.

Description
Creates a TelstraM1Interface instance, initialises the SPI bus and sets up the chip select, reset and data ready pins for communication with the cellular system.

Public Member Functions

int sendCommand(char *buffer, int length)

Parameters

char *buffer	Buffer containing command packet [max 2048 bytes]
int length	Length of command packet in buffer

Return Value
Number of bytes sent.

Description
Sends a packet to the cellular system and reports number of bytes written.

int getResponse(char *buffer, int length, int timeout)

Parameters

char *buffer	Buffer to receive response packet [min 2048 bytes]
int length	Length of response buffer
int timeout	Number of milliseconds to wait for response

Return Value
Number of bytes received (0 if timeout occurred).

Description
Waits for a response from the cellular system, then saves response to buffer and reports number of bytes received. If no response is received within timeout period, returns zero (buffer will be unmodified).

Connection4G Class

Overview

The Connection4G class handles internet connectivity over a CAT-M1 network using the Cellular system. This class is used by the TelstraIoT class to connect to the Telstra IoT platform, but its member functions can also be called directly from Arduino sketches.

Constructor

Connection4G(bool secure, TelstraM1Interface *shield)

Parameters

bool secure	Boolean value indicating whether TLS will be used
TelstraM1Interface *shield	Instance of TelstraM1Interface used to communicate with Cellular system

Return Value

None.

Description

Creates a Connection4G instance. If 'secure' is set to true, TLS will be used to secure all TCP connections made using Connection4G. The Telstra IoT Platform ONLY accepts secure connections so 'secure' must be set to true if this class is to be used with the TelstraIoT class.

Public Member Functions

int ping (const char *host, char *responseBuffer)

Parameters

const char *host	Host name or IP address [max 256 bytes]
char *responseBuffer	Buffer to hold host IP address resolved by Cellular system [min 16 bytes]

Return Value

Response time of host in milliseconds. Returns CONNECTION4G_STATUS_ERROR (-1) either the host or the cellular system did not respond.

Description

Pings the specified host and reports the response time. Also saves the host's resolved IP address to a buffer. If either the host or the cellular system does not respond within 5 seconds, the function will return CONNECTION4G_STATUS_ERROR (-1) and the buffer contents can be considered invalid.

int openTCP(const char *host, uint16_t port)

Parameters

const char *host	Host name or IP address [max 256 bytes]
uint16_t port	Port for TCP connection (typically 80 for HTTP, 443 for HTTPS)

Return Value

Integer status code:
CONNECTION4G_STATUS_OK (0) if connection was successfully established
CONNECTION4G_STATUS_ERROR (-1) if an error occurred or shield was not ready.

Description

Opens a TCP connection to the specified host on the specified port. The connection will be established using TLS if 'secure' was set to true when Connection4G was initialised. Only one TCP connection can be open through the cellular system at any time.

int closeTCP()

Parameters

None.

Return Value

Integer status code:
CONNECTION4G_STATUS_OK (0) if connection was successfully closed
CONNECTION4G_STATUS_ERROR (-1) if the shield did not respond within 15 seconds.

Description

Closes a TCP connection previously opened with openTCP().

int TCPRead(char *buffer, uint16_t length)

Parameters

char *buffer	Buffer to receive data [min 2048 bytes]
uint16_t length	Length of data buffer

Return Value

Returns:
Length of data received (in bytes), if command was successful.
CONNECTION4G_STATUS_ERROR (-1) if no data was received or a timeout occurred.

Description

Listens for incoming data on a TCP connection and saves it to a buffer. If no data is received in 5 seconds or an error occurs, the function will return an error code and the data in the buffer should be considered invalid.

int TCPWrite(char *data, uint16_t length)

Parameters

char *data	Buffer of data to send [max 2048 bytes]
uint16_t length	Length of data buffer

Return Value

Integer status code:
CONNECTION4G_STATUS_OK (0) if data was successfully sent
CONNECTION4G_STATUS_ERROR (-1) if the shield did not respond within 5 seconds.

Description

Sends data to host over a TCP connection.

int getSignalQuality(char *data)

Parameters

char *data	Buffer to save result [min 8 bytes]

Return Value

Integer status code:
CONNECTION4G_STATUS_OK (0) if report was received successfully
CONNECTION4G_STATUS_ERROR (-1) if an error or timeout occurred.

Description

Gets a signal quality report from the 4G modem on the Cellular system. A string will be saved to the data buffer provided containing the current signal level in dBm (this will typically be in the range -120dBm to -50dBm). If no data is received in 5 seconds or an error occurs, the function will return an error code and the data in the buffer should be considered invalid.

int activatePDP(const char *apn, const char *username, const char *password)

Parameters

char *apn	APN to use for PDP context [max 128 bytes]
char *username	Username to use for PDP context [max 128 bytes]
char *password	Password to use for PDP context [max 128 bytes]

Return Value

Integer status code:
CONNECTION4G_STATUS_OK (0) if command was successfully sent to modem
CONNECTION4G_STATUS_ERROR (-1) if an error or timeout occurred.

Description

Activates a new PDP context using the specified APN. Username and password should be empty strings unless login details are required for specified APN.

NOTE: Manually establishing a PDP context using this function is not required for standard use. When powered on, the cellular system will automatically establish a PDP context using Telstra's default APN.

int deactivatePDP()

Parameters
None.

Return Value
Integer status code:
CONNECTION4G_STATUS_OK (0) if command was successfully sent to modem
CONNECTION4G_STATUS_ERROR (-1) if an error or timeout occurred.

Description
Deactivates a PDP previously opened with activatePDP().

TelstraM1Device Class

Overview

The TelstraM1Device class handles communication with the Cellular system and its onboard sensors. This class is used by the TelstraIoT class to generate accurate timestamps for communication with the Telstra IoT Platform, but its member functions can also be called directly from Arduino sketches.

Constructor

TelstraM1Device (TelstraM1Interface *shield)

Parameters

TelstraM1Interface *shield	Instance of TelstraM1Interface used to communicate with Cellular system

Return Value

None.

Description

Creates an TelstraM1Interface instance.

Public Member Functions

void getIMEI(char* resultBuffer)

Parameters

char *resultBuffer	Buffer to hold IMEI [min 16 bytes]

Return Value

None.

Description

Requests the IMEI (International Mobile Equipment Identity) of the Cellular system and saves the result to a buffer.

void getIP(char* resultBuffer)

Parameters

char *resultBuffer	Buffer to hold IP address [min 16 bytes]

Return Value

None.

Description

Requests the public IP address of the Cellular system and saves the result to a buffer.

void getTime(char* resultBuffer)

Parameters

char *resultBuffer Buffer to hold time string [min 50 bytes]

Return Value

None.

Description

Requests the time and date from the real time clock (RTC) on the Cellular system and saves the result to a buffer.

bool isBG96Connected()

Parameters

None.

Return Value

Boolean value indicating CAT M1 modem status.

Description

Returns true if the microcontroller on the cellular system is able to communicate with the modem, false otherwise.

bool isSIMReady()

Parameters

None.

Return Value

Boolean value indicating SIM status.

Description

Returns true if SIM card is inserted and valid, otherwise returns false.

bool isPDPContextActive()

Parameters

None.

Return Value

Boolean value indicating PDP context status.

Description

Returns true if PDP context is active, otherwise returns false.

bool isCellularSystemReady ()

Parameters
None.

Return Value
Boolean value indicating cellular system status.

Description
Returns true if cellular system is ready to accept commands, otherwise returns false.

void waitUntilCellularSystemIsReady ()

Parameters
None.

Return Value
None.

Description
Waits until cellular system is ready to receive commands and has an active, valid PDP context.

void writeCredentials(const char *deviceID, const char *deviceTenant, const char *deviceUsername, const char *devicePassword)

Parameters

const char *deviceID	Device ID string (generated when device is registered on the Telstra IoT Platform). [max 8 bytes]
const char *deviceTenant	Tenant ID for the Telstra IoT Platform. [max 32 bytes]
const char *deviceUsername	Username for the Telstra IoT Platform. [max 32 bytes]
const char *devicePassword	Password for the Telstra IoT Platform. [max 32 bytes]

Return Value
None.

Description
Writes credentials for the Telstra IoT platform to secure, non-volatile memory on the Cellular system. Credentials are saved automatically (using this function) by the registerDevice() function in the TelstraIoT class, so calling it manually is generally not necessary.

void readCredentials(char *deviceID, char *deviceTenant, char *deviceUsername, char *devicePassword)

Parameters

char *deviceID	Buffer to hold device ID string [min 8 bytes]
char *deviceTenant	Buffer to hold tenant ID [min 32 bytes]
char *deviceUsername	Buffer to hold username [min 32 bytes]
char *devicePassword	Buffer to hold password [min 32 bytes]

Return Value
None.

Description
Reads saved credentials from the Cellular system into the buffers provided.

void clearCredentials()

Parameters
None.

Return Value
None.

Description
Deletes saved credentials from the Cellular system .

int getBatteryStatus()

Parameters
None.

Return Value
Integer code representing battery status.
1 - Disconnected
2 - Connected, powering system
3 - Connected, charging

Description
Gets battery status information from the Cellular system.

int getBatteryStateOfCharge()

Parameters
None.

Return Value
Battery state of charge as percentage (0-100)

Description
Gets battery state of charge information from the Cellular system.

void getTemperature(char *temperature)

Parameters

char *temperature Buffer to hold temperature measurement [min 6 bytes]

Return Value

None.

Description

Gets temperature measurement from onboard sensor. Result will be a string containing the temperature in degrees Celsius to two decimal places.

void getLightLevel(char *lightLevel)

Parameters

char *lightLevel Buffer to hold light measurement [min 8 bytes]

Return Value

None.

Description

Gets light level measurement from onboard sensor. Result will be a string containing the measured light level in lux.

void updateRTCFromNetwork(char *resultBuffer)

Parameters

char *resultBuffer Buffer to hold time [min 50 bytes]

Return Value

None.

Description

Updates the onboard real-time clock (RTC) with time from the CAT-M1 network, and saves the current time to the buffer provided. This command is automatically run each time the shield is powered on.

TelstraIoT Class

Overview

The TelstraIoT class handles communication with the Telstra IoT Platform. This high-level class makes use of both the Connection4G and IoTShield classes.

Constructors

TelstraIoT(Connection4G* conn, TelstraM1Device* shield)

Parameters

Connection4G* conn	Instance of Connection4G to use for network connectivity
TelstraM1Device* shield	Instance of TelstraM1Device used to communicate with Cellular system

Return Value

None.

Description

Creates a TelstraIoT instance WITHOUT setting up credentials or hostname. If this constructor is used, the SetCredentials and SetHost functions MUST be called before attempting to connect to the Telstra IoT Platform. This constructor variant is intended for use on a device which has already been registered on the Telstra IoT Platform and has credentials saved to the Cellular system.

TelstraIoT(const char* host, const int port, const char* tenantId, const char* user, const char* password, const char* applicationKey, Connection4G* conn, TelstraM1Device* shield)

Parameters

const char* host	Hostname for Telstra IoT Platform [max 256 bytes]
const int port	Port for connection (typically 443)
const char* tenantId	Tenant ID for Telstra IoT Platform [max 32 bytes]
const char* user	Username for Telstra IoT Platform [max 32 bytes]
const char* password	Password for Telstra IoT Platform [max 32 bytes]
const char* applicationKey	Application Key for Telstra IoT Platform [max 32 bytes]
Connection4G* conn	Instance of Connection4G to use for network connectivity
TelstraM1Device* shield	Instance of TelstraM1Device used to communicate with Cellular system

Return Value

None.

Description

Creates a TelstraIoT instance and sets up hostname, credentials, etc. This constructor variant is best used when registering a new device on the Telstra IoT Platform. Application key can be empty string (“”) if not needed.

TelstraIoT(const char* host, const char* tenantId, const char* user, const char* password, const char* applicationKey, Connection4G* conn, TelstraM1Device* shield)

Parameters

const char* host	Hostname for Telstra IoT Platform [max 256 bytes]
const char* tenantId	Tenant ID for Telstra IoT Platform [max 32 bytes]
const char* user	Username for Telstra IoT Platform [max 32 bytes]
const char* password	Password for Telstra IoT Platform [max 32 bytes]
const char* applicationKey	Application Key for Telstra IoT Platform [max 32 bytes]
Connection4G* conn	Instance of Connection4G to use for network connectivity
TelstraM1Device* shield	Instance of TelstraM1Device used to communicate with Cellular system

Return Value

None.

Description

Creates a TelstraIoT instance and sets up hostname, credentials, etc. Uses default port for HTTPS (port 443). This constructor variant is best used when registering a new device on the Telstra IoT Platform.

Public Member Functions

int raiseAlarm(const char* type, const char* status, const char* severity, const char* time, const char* text)

Parameters

const char* type	Alarm type [max 256 bytes]
const char* status	Alarm status (ACTIVE/ACKNOWLEDGED/CLEARED) [max 32 bytes]
const char* severity	Alarm severity (WARNING/MINOR/MAJOR/CRITICAL) [max 32 bytes]
const char* time	Time of alarm event [max 256 bytes]
const char* text	Text description of alarm event [max 256 bytes]

Return Value

Integer status code:
1 - Alarm message was sent successfully
-2 - Alarm message was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends an alarm notification to the Telstra IoT Platform.

int raiseAlarm(const char* type, const char* status, const char* severity, const char* text)

Parameters

const char* type	Alarm type [max 256 bytes]
const char* status	Alarm status (ACTIVE/ACKNOWLEDGED/CLEARED) [max 32 bytes]
const char* severity	Alarm severity (WARNING/MINOR/MAJOR/CRITICAL) [max 32 bytes]
const char* text	Text description of alarm event [max 256 bytes]

Return Value

Integer status code:
1 - Alarm message was sent successfully
-2 - Alarm message was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends an alarm notification to the Telstra IoT Platform. The timestamp of the alarm notification will be automatically obtained from the RTC on the Cellular system.

int sendMeasurement(const char* type, const char* time, const char* fragmentName, const char* measurementName, const long &mValue, const char* mUnit)

Parameters

const char* type	Measurement type [max 256 bytes]
const char* time	Measurement timestamp [max 256 bytes]
const char* fragmentName	Measurement fragment name [max 256 bytes]
const char* measurementName	Measurement name [max 256 bytes]
const long &mValue	Pointer to long integer containing measurement value
const char* mUnit	Measurement unit [max 256 bytes]

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends an integer measurement to the Telstra IoT Platform.

int sendMeasurement(const char* type, const char* fragmentName, const char* measurementName, const long &mValue, const char* mUnit)

Parameters

const char* type	Measurement type [max 256 bytes]
const char* fragmentName	Measurement fragment name [max 256 bytes]
const char* measurementName	Measurement name [max 256 bytes]
const long &mValue	Pointer to long integer containing measurement value
const char* mUnit	Measurement unit [max 256 bytes]

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends an integer measurement to the Telstra IoT Platform. Timestamp will be automatically obtained from the RTC on the Cellular system.

int sendMeasurement(const char* type, const char* time, const char* fragmentName, const char* measurementName, const long &mValue, const char* mUnit)

Parameters

const char* type	Measurement type [max 256 bytes]
const char* time	Measurement timestamp [max 256 bytes]
const char* fragmentName	Measurement fragment name [max 256 bytes]
const char* measurementName	Measurement name [max 256 bytes]
char* measurementString	String containing measurement [max 256 bytes]
const char* mUnit	Measurement unit [max 256 bytes]

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends a measurement (stored as a string) to the Telstra IoT Platform.

int sendMeasurement(const char* type, const char* fragmentName, const char* measurementName, const long &mValue, const char* mUnit)

Parameters

const char* type	Measurement type [max 256 bytes]
const char* fragmentName	Measurement fragment name [max 256 bytes]
const char* measurementName	Measurement name [max 256 bytes]
char* measurementString	String containing measurement [max 256 bytes]
const char* mUnit	Measurement unit [max 256 bytes]

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Sends a measurement (stored as a string) to the Telstra IoT Platform. Timestamp will be automatically obtained from the RTC on the Cellular system.

int registerDevice(const char* name, char* id, const int idSize, const char** supportedOperations, const int nSupportedOperations, const char** supportedMeasurements, const int nSupportedMeasurements)

Parameters

const char* name	Device name [max 256 bytes]
char* id	Buffer to save device ID [min 8 bytes]
const int idSize	Size of ID buffer
const char** supportedOperations	Array of strings containing list of supported operations [max 512 bytes]
const int nSupportedOperations	Number of supported operations
const char** supportedMeasurements	Array of strings containing list of supported measurement types [max 512 bytes]
const int nSupportedMeasurements	Number of supported measurement types

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Registers a device on the Telstra IoT Platform and sends lists of supported operations and measurements. NOTE: Operations are currently not supported in the TelstraIoT library.

int registerDevice(const char* name, char* id, const int idSize, const char** supportedMeasurements, const int nSupportedMeasurements)

Parameters

const char* name	Device name [max 256 bytes]
char* id	Buffer to save device ID [min 8 bytes]
const int idSize	Size of ID buffer
const char** supportedMeasurements	Array of strings containing list of supported measurement types [max 512 bytes]
const int nSupportedMeasurements	Number of supported measurement types

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Registers a device on the Telstra IoT Platform and sends list of supported measurements.

int registerDevice(const char* name, char* id, const int idSize)

Parameters

const char* name	Device name [max 256 bytes]
char* id	Buffer to save device ID [min 8 bytes]
const int idSize	Size of ID buffer

Return Value

Integer status code:
1 - Measurement was sent successfully
-2 - Measurement was sent but no response was received from the server
Other negative values - HTTP status code of server's response multiplied by negative 1

Description

Registers a device on the Telstra IoT Platform with no supported operations or measurement types.

void setCredentials(const char* _deviceId, const char* _tenantId, const char* _user, const char* _password, const char* _applicationKey)

Parameters

const char* _deviceId	Device ID string [max 8 bytes]
const char* _tenantId	Tenant ID string [max 32 bytes]
const char* _user	Username for Telstra IoT platform [max 32 bytes]
const char* _password	Password for Telstra IoT platform [max 32 bytes]
const char* _applicationKey	Application key for Telstra IoT platform [max 32 bytes]

Return Value
None.

Description
Sets the credentials associated with this TelstraIoT instance. This is necessary if credentials were not provided when instance was initialised.

void setHost(const char* _host, const int _port)

Parameters

const char* _host	Hostname of Telstra IoT Platform [max 256 bytes]
const int _port	Port for connection (typically 443)

Return Value
None.

Description
Sets the hostname and port associated with this TelstraIoT instance. This is necessary if the hostname was not specified when this instance was initialised.

void writeCredentials(const char *deviceID, const char *deviceTenant, const char *deviceUsername, const char *devicePassword)

Parameters

const char *deviceID	Device ID string (generated when device is registered on the Telstra IoT Platform). [max 8 bytes]
const char *deviceTenant	Tenant ID for the Telstra IoT Platform. [max 32 bytes]
const char *deviceUsername	Username for the Telstra IoT Platform. [max 32 bytes]
const char *devicePassword	Password for the Telstra IoT Platform. [max 32 bytes]

Return Value
None.

Description
Writes credentials for the Telstra IoT platform to secure, non-volatile memory on the Cellular system. Credentials are saved automatically (using this function) by the registerDevice() function in the TelstraIoT class, so calling it manually is generally not necessary.

void readCredentials(char *deviceID, char *deviceTenant, char *deviceUsername, char *devicePassword)

Parameters

char *deviceID	Buffer to hold device ID string [min 8 bytes]
char *deviceTenant	Buffer to hold tenant ID [min 32 bytes]
char *deviceUsername	Buffer to hold username [min 32 bytes]
char *devicePassword	Buffer to hold password [min 32 bytes]

Return Value

None.

Description

Reads saved credentials from the Cellular system into the buffers provided. Can be used in conjunction with SetCredentials to load saved credentials and use them with this TelstraIoT instance.

void clearCredentials()

Parameters

None.

Return Value

None.

Description

Deletes saved credentials from the Cellular system.

GPS Class

Overview

The GPS class handles communication with the onboard GPS system. Its member functions can be called directly from Arduino sketches. Locking of GPS might take several minutes for the first time depending on the location of the device.

Constructor

GPS(TelstraM1Interface *shield)

Parameters

TelstraM1Interface *shield	Instance of TelstraM1Interface used to communicate with Cellular system in order to access the onboard GPS

Return Value

None.

Description

Creates a GPS instance.

Public Member Functions

bool enable()

Parameters

NA

Return Value

Boolean status code:

- True if GPS is enabled.
- False if cellular system failed to respond.

Description

Enables onboard GPS system

bool disable()

Parameters

NA

Return Value

Boolean status code:

- True if GPS is disabled.
- False if cellular system failed to respond.

Description

Disables onboard GPS system

Int8_t getStatus()

Parameters

NA

Return Value

1 byte integer status code:

- 1 if GPS is enabled.
- 0 if GPS is disabled
- -1 if cellular system failed to respond.

Description
Returns the current status of the GPS.

GPS_Location * get_Location()

Parameters

NA

Return Value
Address to a struct comprising of GPS location data:

- UTC_time – 12-byte char array
- UTC_date – 8-byte char array
- Latitude – 12-byte char array
- longitude – 12-byte char array
- precision – 6-byte char array
- altitude – 10-byte char array
- positioning_mode – 1-byte unsigned integer
- cog – 8-byte char array depicting course over ground
- speed_kmh – 8-byte char array
- speed_knot – 8-byte char array
- nsat – 1-byte unsigned integer depicting number of satellites locked by the GPS.

Description
Provides GPS location of the device.

TelstraWeb Class

Overview

The TelstraWeb class handles the http/https POST requests. Its member functions can be called directly from Arduino sketches.

Constructor

TelstraWeb(Connection4G* conn, TelstraM1Device* shield)

Parameters

Connection4G* conn	Instance of Connection4G to use for network connectivity
TelstraM1Device* shield	Instance of TelstraM1Device used to communicate with Cellular system

Return Value
None.

Description
Creates a TelstraWeb instance.

Public Member Functions

void setHost(const char* _host, const char* path ,const int _port)

Parameters

const char* _host	Pointer to a host address of destination
const char* _path	Pointer to a path address of destination
const int _port	Port number for the post request

Return Value
None

Description
Setup host destination for the POST request.

int post(const char* data)

Parameters

const char* _data	Pointer to a POST request payload.

Return Value
Integer status code

- -2 if no response received from the cellular system
- -1 if received response code is an invalid HTTP code
- 1 if received response code is a valid HTTP code

Description
Performs the HTTP POST request.

OLED Class

Overview

The OLED class handles the drawing of text and images to the OLED display on the MikroBUS addon. Its member functions can be called directly from Arduino sketches.

Constructor

OLED()

Parameters
None

Return Value
None

Description
Creates an OLED instance.

Public Member Functions

void begin(void)

Parameters
None

Return Value
None

Description
Starts SPI communication and initialises the OLED display.

void Set_OLED_for_write_at_X_Y(uint8_t x, uint8_t y)

Parameters

uint8_t x	x-coordinate of first pixel to be written to display
uint8_t y	y-coordinate of first pixel to be written to display

Return Value
None

Description
Sets up the location of the first OLED pixel to be drawn on the display.

void fill_OLED(uint8_t R, uint8_t G, uint8_t B)

Parameters

uint8_t R	Red colour value
uint8_t G	Green colour value
uint8_t B	Blue colour value

Return Value

None

Description
Sets every OLED pixel to the chosen colour.

void drawPixel(uint16_t x, uint16_t y, uint8_t R, uint8_t G, uint8_t B)

Parameters

uint16_t x	x-coordinate of pixel
uint16_t y	y-coordinate of pixel
uint8_t R	Red colour value
uint8_t G	Green colour value
uint8_t B	Blue colour value

Return Value
None

Description
Sets the OLED pixel at location (x,y) to the chosen colour.

void drawCircle(uint16_t x0, uint16_t y0, uint16_t radius, uint16_t R, uint16_t G, uint16_t B)

Parameters

uint16_t x0	x-coordinate of centre of circle
uint16_t y0	y-coordinate of centre of circle
uint16_t radius	Radius of circle
uint8_t R	Red colour value
uint8_t G	Green colour value
uint8_t B	Blue colour value

Return Value
None

Description
Draws a circle of desired radius in the chosen colour. The centre is located at (x0,y0).

void fastHorizontalLine(uint16_t x0, uint16_t y, uint16_t x1, uint8_t R, uint8_t G, uint8_t B)

Parameters

uint16_t x0	Initial x-coordinate of line
uint16_t y	y-coordinate of line
uint16_t x1	Final x-coordinate of line
uint8_t R	Red colour value
uint8_t G	Green colour value
uint8_t B	Blue colour value

Return Value
None

Description
Draws a line between pixels (x0,y) and (x1,y) (inclusive) in the chosen colour.

void drawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint8_t r, uint8_t g, uint8_t b)

Parameters

uint16_t x0	Initial x-coordinate of line
uint16_t y0	Initial y-coordinate of line
uint16_t x1	Final x-coordinate of line
uint16_t y1	Final y-coordinate of line
uint8_t r	Red colour value
uint8_t g	Green colour value
uint8_t b	Blue colour value

Return Value
None

Description
Draws a line between pixels (x0,y0) and (x1,y1) (inclusive) in the chosen colour.

void fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint8_t r, uint8_t g, uint8_t b)

Parameters

uint16_t x	Initial x-coordinate of rectangle
uint16_t y	Initial y-coordinate of rectangle
uint16_t w	Width of rectangle
uint16_t h	Height of rectangle
uint8_t r	Red colour value
uint8_t g	Green colour value
uint8_t b	Blue colour value

Return Value
None

Description
Draws solid rectangle of width ‘w’ and height ‘h’ with top left corner at (x,y) in the chosen colour.

void drawChar(unsigned char c, int16_t x, int16_t y, uint8_t r, uint8_t g, uint8_t b, uint8_t size)

Parameters

unsigned char c	Character to be drawn to display
uint16_t x	Top left x-coordinate
uint16_t y	Top left y-coordinate
uint8_t r	Red colour value
uint8_t g	Green colour value

uint8_t b	Blue colour value
uint8_t size	Scaling factor of font

Return Value
None

Description
Draws a single character to the display with top left position at (x,y). The character can be resized.

void drawString(String str, int16_t x, int16_t y, uint8_t r, uint8_t g, uint8_t b, uint8_t size)

Parameters

String str	String to be drawn to display
uint16_t x	Top left x-coordinate
uint16_t y	Top left y-coordinate
uint8_t r	Red colour value
uint8_t g	Green colour value
uint8_t b	Blue colour value
uint8_t size	Scaling factor of font

Return Value
None

Description
Draws a string of characters to the display with top left position at (x,y). The characters can be resized.

void drawImage(const uint8_t* img, int16_t x, int16_t y, int16_t w, int16_t h)

Parameters

const uint8_t* img	Pointer to image data location
uint16_t x	Top left x-coordinate
uint16_t y	Top left y-coordinate
uint16_t w	Width of image
uint16_t h	Height of image

Return Value
None

Description
Draws an image to the display with top left corner located at (x,y). The data is in the format of the payload of a 24-bit bitmap image.

END OF DOCUMENT