afLib2 for Arduino API

afLib2 for Arduino is an Arduino library that can be used with Arduino 1.6 and later to communicate with an Afero radio module over serial interfaces such as SPI and UART. The library is written in C and provides a simple, lightweight API that provides all the functionality you should need to implement your application. You can download the library from http://github.com/aferodeveloper/afLib2.

If you are transitioning from C++ afLib to C afLib, please refer to the afLib to afLib2 Migration Guide.

The library includes the following functions:

- arduino_transport_create_<*protocol*>()
- af_lib_create()
- af_lib_get_attribute()
- af_lib_loop()
- af_lib_set_attribute()

We also describe two callback functions you should write to work with afLib2:

- attrSetHandler()
- attrNotifyHandler()

If you need code examples that use the deprecated C++ afLib, download the archive file (.zip).

arduino_transport_create_<*protocol*>()

Description

Create an instance of `af_transport_t` appropriate for the communications protocol to be used by the application. A pointer to the `af_transport_t` is a required argument for `af_lib_create()`.

There is a specific form of `af_transport_t` for each of the communication protocols available, SPI and UART. You will need to create the `af_transport_t` corresponding to the protocol used in your project, and pass it to `af_lib_create`. For example code, see af_lib_create().

SPI Syntax

```
af_transport_t* arduino_transport_create_spi(int chipSelect);
```

Parameters for arduino_transport_create_spi

`chipSelect`	The number of the pin to be used for SPI Chip Select. It is 10 for Teensy and Uno.

UART Syntax

```
af_transport_t* arduino_transport_create_uart(uint8_t rxPin, uint8_t txPin)
```

Parameters for arduino_transport_create_uart

`rxPin`	The number of the pin to be used for UART RX. It is 7 for Teensy and Uno.
`txPin`	The number of the pin to be used for UART TX. It is 8 for Teensy and Uno.

Returns

Pointer to an `af_transport_t`.

af_lib_create()

Description

Create an instance of afLib2 for the application. The parameters include some callbacks to let your code know when things have changed, and a pointer to the transport specific to your communication protocol.

Syntax

```
af_lib_t* af_lib_create(attr_set_handler_t attr_set,
                        attr_notify_handler_t attr_notify,
                        af_transport_t *the_transport)
```

Parameters

`attr_set`	The callback (`attributeSetHandler()`) to request your MCU change one of its local attributes.
`attr_notify`	The callback (`attributeNotifyHandler()`) to let your MCU code know that one of the ASR attributes has changed.
`the_transport`	Pointer to an `af_transport_t` to handle communications.

Returns

Pointer to an instance of afLib2 to handle communications.

SPI Example

The SPI example below shows how we create a new instance of afLib2 for SPI:

- Instantiate an `af_transport_t` for SPI, supplying the pin number for Chip Select.
- Supply `af_transport_t` as an argument to `af_lib_create()`, along with `attrSetHandler` and `attrNotifyHandler` as the two callback methods.

- Call `arduino_spi_setup_interrupts` to give afLib2 access to interrupts.

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"

// Pin Defines assume Arduino Uno
#define CS_PIN                    10
#define INT_PIN                    2

af_lib_t *af_lib;

bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
  // Any application-specific actions
  return true;
}

void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {
  // Any application-specific actions
}

void setup() {
  // Initialize afLib
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib); in sketch's loop()
  af_lib_loop(af_lib);
}
```

UART Example

In the UART example, we create a new instance of afLib2 for UART. In this case we:

- Instantiate an `af_transport_t` for UART, supplying RX and TX pin numbers.
- Call `af_lib_create()`, with `attrSetHandler` and `attrNotifyHandler` as the callback methods and the UART transport pointer.

- Include `ArduinoUART.h`, and and specify `ARDUINO_USE_UART` (by uncommenting) in `arduino_transport_selection.h`.

```
#include "af_lib.h"
#include "arduino_uart.h"

#define RX_PIN                7
#define TX_PIN                8

af_lib_t *af_lib;

bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
  // Any application-specific actions
  return true;
}

void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {
  // Any application-specific actions
}

void setup() {
  // Initialize afLib
  af_transport_t *arduinoUART = arduino_transport_create_uart(RX_PIN, TX_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoUART);
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib); in sketch's loop()
  af_lib_loop(af_lib);
}
```

The code examples for the functions below illustrate the use of SPI communication. To use UART instead, only the minimal changes in afLib2 instantiation, as highlighted above, would be required.

af_lib_get_attribute()

Description

Request the value of a specified attribute. The `af_lib_get_attribute()` call only queues the request; your attrNotifyHandler() function will be called when the value is returned by ASR. The attribute may or may not have a default value defined in the Profile Editor.

When called:

- If no default value has been set, `af_lib_get_attribute` will return `length=0`, `value=NULL`.
- If the default value has been set and the attribute has not been modified, `af_lib_get_attribute` will return the default value.
- If the attribute has been modified by the service or MCU, `af_lib_get_attribute` will return the most recent value set by either the service or the MCU.

Syntax

```
af_lib_get_attribute(af_lib_t *af_lib, const uint16_t attr_id)
```

Parameters

`af_lib`	Pointer to the active aflib instance.
`attr_id`	The attribute ID of the value you are requesting.

Returns

A result code indicating whether request was queued successfully.

Example

In the example below, `af_lib_get_attribute()` is called in `loop()` to obtain the value of `AF_MY_ATTRIBUTE`. If the call returns success, we set `shouldGetAttr` to false, since we won't need to call `af_lib_get_attribute()` again. The value of the attribute of interest is actually obtained when `attrNotifyHandler()` is called. In that callback, we can use the attribute ID argument to select which action should be taken (in this case, we simply print to console).

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "profile/device-description.h"

// Pin Defines assume Arduino Uno
#define CS_PIN               10
#define INT_PIN               2

boolean shouldGetAttr = true;
af_lib_t *af_lib;

bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
  // Any application-specific actions
  return true;
}

void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {
  if (attributeId == AF_MY_ATTRIBUTE) {
        Serial.print("*attrNotifyHandler*        Attr id: ");
        Serial.print(attributeId);
        Serial.print(" value: ");
        Serial.println(*value);
  }
}

void setup() {
  arduino_logger_start(115200);

  // Initialize afLib
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  if (shouldGetAttr) {
    // Attribute name definitions are provided by device-description.h from your profile
    int result = af_lib_get_attribute(af_lib, AF_MY_ATTRIBUTE);
    if (result == AF_SUCCESS) {
      shouldGetAttr = false;
    }
```

```
    else {
      Serial.print("*af_lib_get_attribute*       return error: ");
      Serial.println(result);
    }
  }

  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

af_lib_loop()

Description

Give afLib2 processing time.

Syntax

`void af_lib_loop(af_lib_t *af_lib)`

Parameters

`af_lib`	Pointer to the active aflib instance.

Returns

None.

Example

The `loop()` example serves to reiterate that `af_lib_loop(af_lib)` should be called within the `loop()` method of your MCU code, to allow the afLib2 state machine to run, and call `attrSetHandler` and `attrNotifyHandler` callbacks when appropriate.

```
af_lib_t *af_lib;

void setup() {
  // Initialize afLib
```

```
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

af_lib_set_attribute()

Description

Request to change the value of a specified ASR attribute. This call only queues the request. Your attrNotifyHandler() function will be called when the value has been changed by ASR.

Syntax

There are several versions of the `af_lib_set_attribute` API that allow you to pass values of different byte-lengths to afLib2. Here are the currently supported types:

```
int af_lib_set_attribute_bool(af_lib_t *af_lib, const uint16_t attr_id, const bool value);
int af_lib_set_attribute_8(af_lib_t *af_lib, const uint16_t attr_id, const int8_t value);
int af_lib_set_attribute_16(af_lib_t *af_lib, const uint16_t attr_id, const int16_t value);
int af_lib_set_attribute_32(af_lib_t *af_lib, const uint16_t attr_id, const int32_t value);
int af_lib_set_attribute_64(af_lib_t *af_lib, const uint16_t attr_id, const int64_t value);
int af_lib_set_attribute_str(af_lib_t *af_lib, const uint16_t attr_id, const uint16_t
value_len, const char *value);
int af_lib_set_attribute_bytes(af_lib_t *af_lib, const uint16_t attr_id, const uint16_t
value_len, const uint8_t *value);
```

Parameters

`af_lib`	Pointer to the active aflib instance.
`attr_id`	The attribute ID for which you are requesting the value.
`valueLen`	The size in bytes of the attribute value. This parameter only needed when value size is not

	indicated by value type.
`value`	The new value for the attribute.

Returns

A result code indicating whether request was queued successfully.

Example

Within `loop()`, we call `af_lib_set_attribute()` to set the value of the GPIO 1 attribute of ASR. ASR will change the value of the GPIO attribute to 1, set the level of the GPIO appropriately, then call the MCU's `attrNotifyHandler()` method, allowing us perform any actions desired once we know the operation is complete.

Notice that in this example we're including the device-description header (`"profile/device-description.h"`) generated by the Afero Profile Editor. When we used the Profile Editor to define the attributes for the example application, attribute names such as `AF_GPIO_1` were written to this header for use in MCU code.

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "profile/device-description.h"

// Pin Defines assume Arduino Uno
#define CS_PIN               10
#define INT_PIN               2

boolean shouldSetAttr = true;
uint8_t value = 1;
af_lib_t *af_lib;

bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
  // Any application-specific actions

  return true;
```

```
}

// attrNotifyHandler() is called when either an ASR module attribute has been changed or in
response to a af_lib_get_attribute operation
// Add any actions required by your specific application in those contexts
void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {

  // Any application-specific actions
  if (attributeId == AF_GPIO_1) {
    Serial.print("*attrNotifyHandler*   Attr id: ");
    Serial.print(attributeId);
    Serial.print(" value: ");
    Serial.println(*value);
  }
}

void setup() {
  arduino_logger_start(115200);

  // Initialize afLib
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  if (shouldSetAttr) {
        // Attribute name definitions are provided by device-description.h from your profile
        int result = af_lib_set_attribute_16(af_lib, AF_GPIO_1, value);
        if (result == AF_SUCCESS) {
          shouldSetAttr = false;
        }
        else {
          Serial.print("*set_attribute*  return error: ");
          Serial.println(result);
        }
  }

  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

attrSetHandler()

Description

Callback that allows ASR to request an MCU attribute be changed. You should define this function in your MCU firmware to perform application-specific actions your code must take (e.g., updating the state of the hardware), in light of the attribute value change.

Syntax

```
bool attrSetHandler(const uint8_t request_id,
                    const uint16_t attribute_id,
                    const uint16_t value_len,
                    const uint8_t *value);
```

Parameters

`request_id`	The request ID for this request.
`attribute_id`	The ID of the attribute that is to be set.
`value_len`	The size in bytes of the attribute value.
`value`	The new value of the attribute.

Returns

- `True` if the host MCU was able to update the attribute internally.
- `False` if the host MCU was unable.

Example

`attrSetHandler()` is called by afLib2 to allow us to update the device state to correspond to the change in attribute value. In this specific example, we change the LED state in response to an attribute change (which may have been triggered, for example, by a user action in the mobile application UI).

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "profile/device-description.h"

// Pin Defines assume Arduino Uno
#define CS_PIN                10
#define INT_PIN                2

af_lib_t *af_lib;

// attrSetHandler() is called when a client changes an attribute;
// define this callback to include any MCU actions that should be performed in that context.
bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
  Serial.print("*attrSetHandler*         id: ");
  Serial.print(attributeId);
  Serial.print(" value: ");
  Serial.println(*value);

  // Any application-specific actions required to update the actual device state to correspond
to the attribute value
  if (attributeId == AF_LED_ATTR) {
        digitalWrite(LED1_PIN, *value);
  }

  // Return value from this call informs service whether or not MCU was able to update local
state to reflect the
  // attribute change that triggered callback.
  // Return false here if your MCU was unable to update local state to correspond with the
attribute change that occurred;
  // return true if MCU was successful.
  return true;
}

void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {
  // Any application-specific actions
}

void setup() {
  arduino_logger_start(115200);

  pinMode(LED1_PIN, OUTPUT);
```

```
  // Initialize afLib
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

attrNotifyHandler()

Description

Application callback that allows afLib2 to notify that an attribute has changed. This method will be called in response to an `af_lib_get_attribute` call from the application and whenever an ASR module attribute changes.

Syntax

```
void attrNotifyHandler(const uint8_t request_id,
                       const uint16_t attribute_id,
                       const uint16_t value_len,
                       const uint8_t *value);
```

Parameters

`request_id`	The request ID for this request.
`attribute_id`	The ID of the attribute that has changed.
`value_len`	The size in bytes of the attribute value.
`value`	The new value for the attribute.

Returns

None.

Example

`attrNotifyHandler()` will be called by afLib2 in response to either `af_lib_get_attribute()` or an attribute change on ASR. If in response to an `af_lib_get_attribute()` call, the MCU code will have the attribute value (via argument passed with `attrNotifyHandler()`). If due to an attribute's changing state on ASR, the MCU code can use the passed-in `attributeId` to determine which attribute changed and respond appropriately. In the example above we simply print passed-in parameters.

```
#include <SPI.h>
#include "af_lib.h"
#include "arduino_spi.h"
#include "profile/device-description.h"

// Pin Defines assume Arduino Uno
#define CS_PIN                10
#define INT_PIN                2

af_lib_t *af_lib;

bool attrSetHandler(uint8_t requestId, const uint16_t attributeId, const uint16_t valueLen,
const uint8_t *value) {
    // Any application-specific actions
}

// attrNotifyHandler() is called when either an ASR module attribute has been changed or in
response to a af_lib_get_attribute operation.
// Add any actions required by your specific application in those contexts.
void attrNotifyHandler(const uint8_t requestId, const uint16_t attributeId, const uint16_t
valueLen, const uint8_t *value) {
  Serial.print("*attrNotifyHandler*      Attr id: ");
  Serial.print(attributeId);
  Serial.print(" value: ");
  Serial.println(*value);
}

void setup() {
  arduino_logger_start(115200);
```

```
  pinMode(LED1_PIN, OUTPUT);

  // Initialize afLib
  af_transport_t *arduinoSPI = arduino_transport_create_spi(CS_PIN);
  af_lib = af_lib_create(attrSetHandler, attrNotifyHandler, arduinoSPI);
  arduino_spi_setup_interrupts(af_lib, digitalPinToInterrupt(INT_PIN));
}

void loop() {
  // Give afLib processing time by calling af_lib_loop(af_lib) in sketch's loop()
  af_lib_loop(af_lib);
}
```

See Also

af_lib_get_attribute()

Next: *afLib to afLib2 Migration Guide*

Updated June 6, 2018