Iris E-course
Innovation through Education

Introduction to the Iris Robot Platform
Appendix B: Iris Library

Presented By:

Higher Order Innovation

www.higherorderinnovation.com

Updated January 15, 2022

Appendix B: Iris Library

B.1 Adding an Arduino Library through the Library Manager

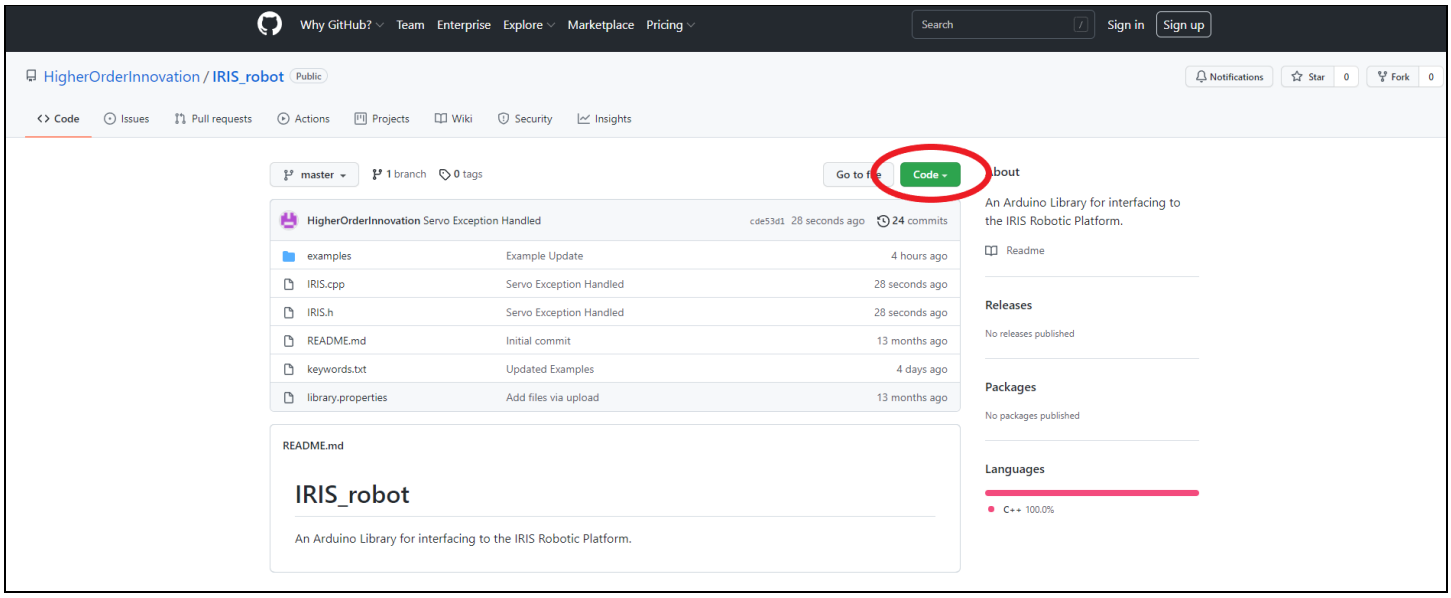
Large libraries can be found through the integrated library manager utility within the Arduino IDE. This utility can be found through Sketch->Include Library->Library Manager. Once there, it is important to note a few things.

1. Not all versions of a particular library may be compatible. While most libraries try to be backwards compatible when possible, sometimes, library upgrades can break your code. Keeping an eye on the installed version of a particular library and the one called out in any ebook or guide is important to prevent odd errors from showing up.
2. Library names are not universally unique. For instance, a library like IRremote.h seems like it should work regardless of what is downloaded, but each can have different interfaces and methods that it uses. Just as the version number matters to make code run properly, so does the authorship of your library.

B.2 Adding a Library from Github

Some larger libraries are accessible through the Library Manager explored in the previous section. When a library is not built into this system in Arduino's IDE, the library can easily be added through its Github repository. A Github repository is a very common system of source control and allows public access to developed code. In the case of Iris, the primary usage of the Github process is used for the main Iris Library, but the same method can be used for additional libraries.

1. Find the repository. For example https://github.com/HigherOrderInnovation/IRIS_robot is the link to the main Iris Library. From that page you can select the Green button and select to download the .zip copy of the library.

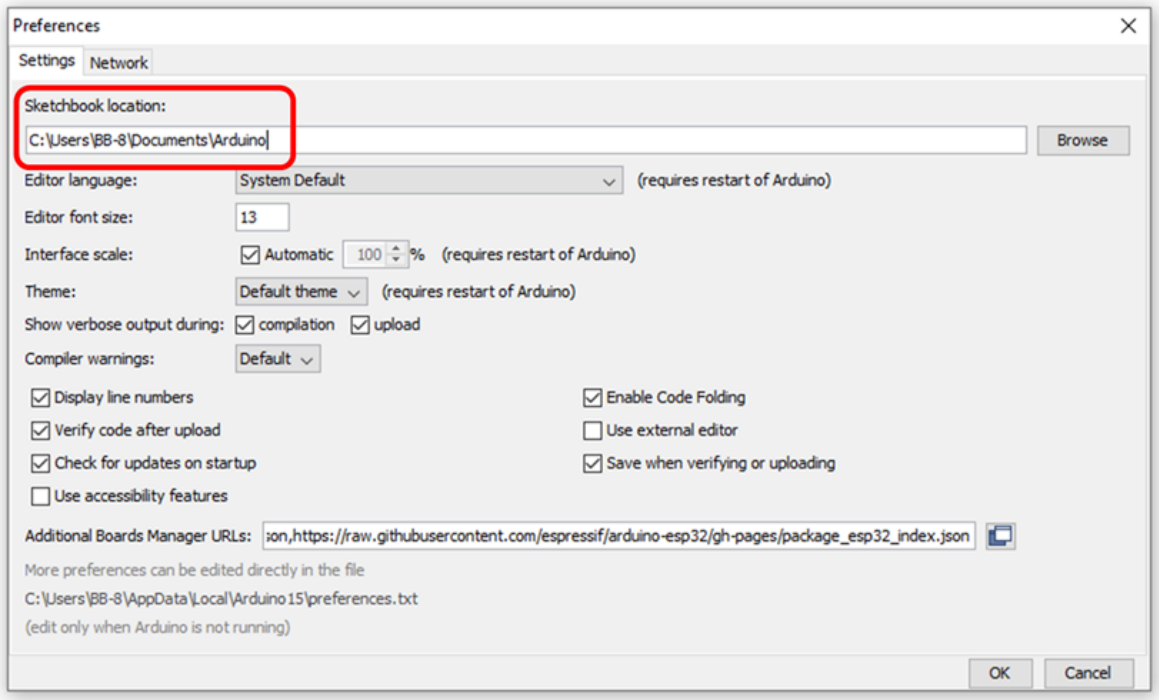


2. Within the Arduino IDE, go to Sketch->Include Library->Add .ZIP Library. In the window that opens, direct it towards the .zip file that you just downloaded. On most computers, this will be found in the download folder of your computer.
3. If you have previously installed the same library through this method, unfortunately, this new copy of the library will not upgrade the existing one by default. To do this, you must first manually delete the first version. For instructions on deleting the previous version, see the next section. If this is the first time you have downloaded the library, congratulations, you are ready to go.

B.3 Removing a Library Installed in Arduino

Sometimes, removal of a library is necessary to upgrade to a new version or to simply keep your installed libraries cleaned up. To do this takes a little more maneuvering through your computer as it is not built into the IDE, but steps are not too involved and you can follow along below.

1. Within Arduino, go to File->Preferences and take note of the file directory of your “Sketchbook Location” as boxed below.



2. Using your computer's File Explorer (Windows machines), go to the location from your preferences window. Within that folder, you will find a folder called libraries.
3. Open the libraries folder and find the library you would like to remove from Arduino. In the case of upgrading the Iris library, find the folder named Iris_robot-master and delete it.
4. Now you can go through the steps to add the newly downloaded Iris library with the steps in the previous section.

B.4 Iris Library Overview

The Iris library was built to allow easier connection and usage of the Iris Robot Platform. Once the built-in features of the Iris robot are mastered, expanded capabilities can be utilized through the addition of the Iris dock. The dock expands the reach of Iris to include external motors, a larger battery, as well as external sensors.

Iris Robot Functions

void setLED(bool state);	Turn the center LED on/off
int getLightReading();	Read how much light is hitting the front sensor
int getNeoPin();	Retrieve the pin that the NeoPixels are connected to
int getIrPin();	Retrieve the pin that the Infrared receiver is connected to
bool getButtonState();	Read the state of the pushbutton
int getButtonPin();	Retrieve the pin that the button is connected to
void setMotors(int left, int right);	Command the speed and direction of the on-board motors

Iris Dock Functions

attachDockMotor(int position);	Startup a 3-wire motor
setDockMotor(int position, int speed);	Command a 3-wire motor
SmartMotorPosition(HardwareSerial &SerialX, uint8_t id, int16_t position, uint16_t time);	Command a Smart Motor to a specific position
void SmartMotorStopMove(HardwareSerial &SerialX, uint8_t id);	Command a Smart Motor to stop moving
void SmartMotorSetID(HardwareSerial &SerialX, uint8_t old_id, uint8_t new_id);	Command a Smart Motor to reassign its ID value
void SmartMotorSpin(HardwareSerial &SerialX, uint8_t id, int16_t speed);	Command a Smart Motor to Spin at a certain speed
int SmartMotorReadPosition(HardwareSerial &SerialX, uint8_t id);	Command a Smart Motor to respond with its angular position
int SmartMotorReadVin(HardwareSerial &SerialX, uint8_t id);	Command a Smart Motor to respond with the voltage of the power supply

B.5 Iris Robot Functions

The functions within this section allow interaction and interfacing to the components on every Iris robot. Access to the NeoPixels, LED, pushbutton, infrared receiver, light sensor, and motors can be done through or with the help of the following functions.

Function

void setLED(bool _state);

Description

Control the state (ON or OFF) of the central blue LED on the Iris robot

Parameter

_state *true* turns on the LED, *false turns off the LED*

Return

Nothing is returned

Example

```
setLED(true); // Turns the center LED on
setLED(false; // Turns the center LED off
```

Function

int getLightReading();

Description

Read the value from the light sensor on the front of the robot

Parameter

None

Return

An integer value from 0 to 4095 is returned. Lower values for darkness and high values for light.

Example

```
int lightLevel = getLightReading();
    // Read the light value from the sensor and save that value to the variable lightLevel
```

Function

int getNeoPin();

Description

Retrieve the pin that can control the NeoPixels

Parameter

None

Return

An integer value of 18

Example

See Ch06-NeoPixel example program for its usage.

Function

int getIrPin();

Description

Retrieve the pin that can receive the Infrared communication

Parameter

None

Return

An integer value of 23.

Example

See Ch09-IR example program for its usage.

Function

bool getButtonState();

Description

Read the state of the pushbutton

Parameter

None

Return

A boolean value of *true* (pressed) or *false* (released)

Example

```
if(robot.getButtonState()){ // If the button is pressed
        // Do something }
```

Function

bool getButtonPin();

Description

Retrieve the pin that is connected to the button

Parameter

None

Return

An integer value of 15

Example

```
int pushButtonPin = robot.getButtonPin();
    // Get the button Pin and store its value in the variable pushButtonPin.
```

Function

void setMotors(int left, int right)

Description

Command the left and right motors to turn.

Parameters

Left An integer value from -100 to 100, where a value of 0 stops the motor..

Right An integer value from -100 to 100, where a value of 0 stops the motor,

Return

None

Example

```
robot.setMotors(-100, -100); // Drive backwards at full speed
robot.setMotors(50,-50); // Turn clockwise half speed
robot.setMotors(0,100); // Hold the left wheel from spinning and pivot the robot around that point.
```

B.6 Iris Dock Functions

When expanding beyond the base Iris robot, the dock enables more complex robots to be built. In order to engage with the two different types of supported motors as well as the available sensor ports, the following library functions were included in the base Iris Library.

3-wire Motors

The bank of 3-wire ports nearest the battery input is designed for 3-wire PWM-driven motors.

Function

Void attachDockMotor(int position);

Description

Before commanding a 3-wire motor, the code needs to know that a motor has been plugged in.

Parameters

position The position (1-6) that the motor is plugged into. Motor Port 1 is furthest away from Iris.

Return

None

Example

attachDockMotor(3); // Indicate that a 3-wire motor is plugged into Motor Port 3

Function

Void setDockMotor(int position, int speed);

Description

Command the motor to either spin with a certain speed

Parameters

Int position The position (1-6) that the motor is plugged into. Motor Port 1 is furthest away from Iris.
Int speed A value from -100 to 100 where an integer value of 0 is stopped.

Return

None

Example

setDockMotor(3, 50); // Command the motor in position 3 to turn at half speed.

Smart Digital Motors

The smart digital motors can be plugged into either of the two available white connectors on the board. These motors require a little more to set up, but offer great flexibility in design.

Function

void SmartMotorPosition(HardwareSerial &SerialX, uint8_t id, int16_t position, uint16_t time);

Description

Send a motor to a particular angular position.

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t id	The user-assigned ID number of the target motor
uint16_t position	The target position that the motor will go to. 0….1000
uint16_t time	How many milliseconds to spend moving to that position

Return

None

Example

```
SmartMotorPosition(robotSerial, 14, 750, 500);
    // While communicating through robotSerial, command motor number 14 to go to position 750 in 500 milliseconds
```

Function

void SmartMotorStopMove(HardwareSerial &SerialX, uint8_t id);

Description

Stop a particular motor from turning

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t id	The user-assigned ID number of the target motor to stop

Return

None

Example

```
SmartMotorStopMove(robotSerial, 14);
    // While communicating through robotSerial, command motor number 14 to stop moving
```

Function

void SmartMotorSetID(HardwareSerial &SerialX, uint8_t old_id, uint8_t new_id);

Description

Assign a new ID number to a connected motor.

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t old_id	The old ID of the motor that should change. 0 - 254 are valid ID numbers.
uint8_t new_id	The new ID to assign to the motor. 0-254 are valid ID numbers.

Note: An ID of 255 will communicate with all motors connected at that time. Use this as the old_id when assigning to an unknown motor. Ensure that only the target motor is installed during that operation.

Return

None

Example

```
SmartMotorSetID(robotSerial, 14, 21);
    // While communicating through robotSerial, command Motor 14 to be reassigned as motor 21

SmartMotorSetID(robotSerial, 255, 21);
    // While communicating through robotSerial, command all motors connected to be assigned ID of 21
```

Function

void SmartMotorSpin(HardwareSerial &SerialX, uint8_t id, int16_t speed);

Description

Command a particular motor to spin.

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t id	The user-assigned ID number of the target motor to stop
uint16_t speed	A value between -100 and 100 corresponding to speed where a value of 0 is the stop command

Return

None

Example

```
SmartMotorSpin(robotSerial, 14, 75);
    // While communicating through robotSerial, command motor number 14 to spin at 75% of full speed
```

Function

int SmartMotorReadPosition(HardwareSerial &SerialX, uint8_t id);

Description

Read the current angular position of a particular motor

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t id	The user-assigned ID number of the target motor to read

Return

An integer value relating to the motor's current position.

Example

```
int armAngle = SmartMotorReadPosition(robotSerial, 14);
    // While communicating through robotSerial, read the current position of motor 14 and save the value to the variable armAngle
```

Function

int SmartMotorReadVin(HardwareSerial &SerialX, uint8_t id);

Description

Read the current motor power supply voltage level

Parameters

HardwareSerial &SerialX	The Serial communication object created to communicate with the motors.
uint8_t id	The user-assigned ID number of the target motor to read

Return

An integer value relating to the motor's current power supply level.

Example

```
int batteryLevel = SmartMotorReadVin(robotSerial, 14);
    // While communicating through robotSerial, read the current power level feeding motor 14 and save the value to the variable batteryLevel
```

Sensor Ports

Six sensor ports are available on the Dock. These ports can be accessed through the Arduino-built in functions of at least digitalWrite(), digitalRead(), and analogRead(). See the included examples for more detailed usage.