March 19th, 2015

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, BC
V5A 1S6

Re: ENSC 440 Design Specifications - Hermes: Camera Slider Motion Controller

Dear Dr. Rawicz,

Attached is the functional specifications report for our camera slider solution for our Capstone project. Our project is the Hermes camera slider which is a tool that can be used by film producers to make steady time lapse photography. The objective of this project is to produce a solution to control an existing product, which is user friendly and affordable.

Two phases of products are covered in the design specifications for the Hermes camera slider: proof of concept, and pre-production. Moreover, this document contains system overview, functionality constraints, general requirements as well as sustainability and safety of our design.

The founding partners of our company, Behnaz Edalat, Brendan Keane, Calvin Scott, Justin Raine and Martin Palibroda would like to personally thank you for your interest in our functional specifications report. For any reason, feel free to contact us at vitamotu.hermes@gmail.com.

Sincerely Yours,

Edalat

Behnaz Edalat,
vitaMotu Ltd.

ENSC 305W/440

Design Specification

February 16, 2015

Behnaz Edalat	301127510
Brendan Keane	301167176
Martin Palibroda	301179246
Justin Raine	301041695
Calvin Scott	301169228

Abstract

This document describes the design specifications and details for the Hermes camera slider controller along with the individual components used in the final product. A detailed look at each portion of this design - including the hardware, software, and firmware - is the goal of this document, in order to help readers with an overview of the design justifications.

Our project consists of implementing a mobile application to wirelessly communicate with a microcontroller in order to steer the motor and load across an existing camera slider. The distinctive factor of the Hermes is the mobile application that anyone could download to their smartphones and which could be upgraded with new details and features, while also keeping the overall cost of the product below market standard.

This design document focuses more on the technical details of hardware, software, and firmware components along with justification, proof, and argument of chosen parts. A descriptive, designed test plan is also presented in order to test different components of the Hermes motion controller system, and to ensure success of the project.

Table of Contents

List of Figures

List of Tables

Glossary

Term	Definition
App	Shorthand for mobile application
Time-lapse	Long exposure
Stop-motion	Discrete motion with image captures
IDE	Integrated Development Environment
MCU	Microcontroller Unit
AVR	Modified Harvard Architecture Class of Microcontroller
GCC	GNU Compiler Collection
GNU	“GNU’s Not Unix!”
RPM	Rotations Per Minute
iOS	iPhone Operating System

1.0 Introduction

The Hermes camera motion controller is an innovative, economical, and functional solution to capturing steady, quality time-lapse and stop-motion recordings. In cinematography and video production, smooth motion is the key to cinematic and natural looking shots. The Hermes is a mobile application based controller will allow camera to move smoothly along the slider. Figure 1 shows the camera slider we will be using in our development process.

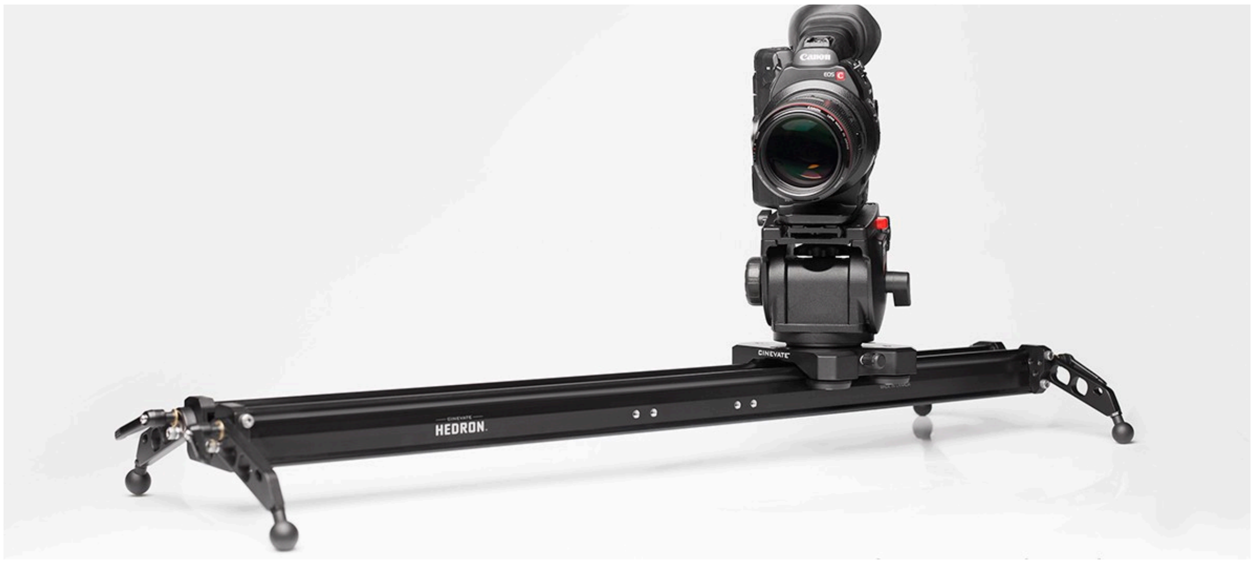


Figure 1: Cinevate Hedron Slider System [7]

The vitaMotu Hermes™ is an upgrade for independent videographers and professional filmmakers who add remote motion control for mounted cameras, interfaced through Wifi connected iPhone app. The Hermes will be an add-on to existing slider systems on the market and control app could be easily downloaded on Smartphones. With a target MSRP $1000, the Hermes will provide an exceptional feature set at a remarkable price.

Straightforward motor and controller installation and downloading app on Smartphones make Hermes user friendly and easy to operate. Furthermore, additional and effective features could be added to mobile application later based on users demands.

1.1 Scope

This design specification supports the previous *Functional Specification* document, by outlining the functional requirement of Hermes camera slider controller. This document describes design specifications for the motor, the motor controller, the control box, the wireless communication, and the mobile application along with the enclosure design and test plan. This document will justify the design decisions and it's a template for future modifications.

1.2 Intended Audience

This document is intended to be used both internally by the team at vitaMotu to implement the design as well as track progress; additionally it will be used externally by stakeholders to understand the implementation of the Hermes camera slider.

2.0 System Specifications and Justification

This section details the basic system specification and descriptions, as well as the justification for the Hermes slider system. The Hermes camera motion controller consists of two main systems. The first of these systems is the motor system, which consists of the Arduino Uno, a motor shield, Bluetooth communication chip, 12V stepper motor, and 1-inch diameter gear system and motor mount. The second of these systems is the mobile application, which allows the user to either directly control the unit in a live mode or set parameters that do long stop-motion and time-lapse modes. In addition, the application allows users to set system wide parameters such as the max speed of the motor and varying amounts of time for crossing the camera slider.

The following figure outlines the high-level block diagrams of the motor control block and phone control block.

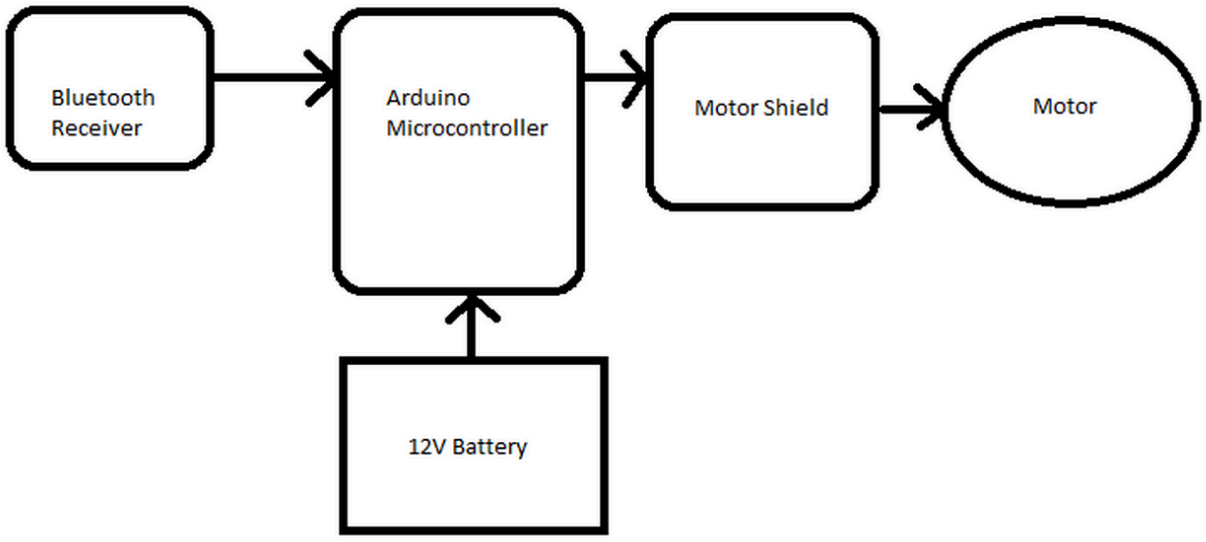


Figure 2: Block Diagram of Motor Unit

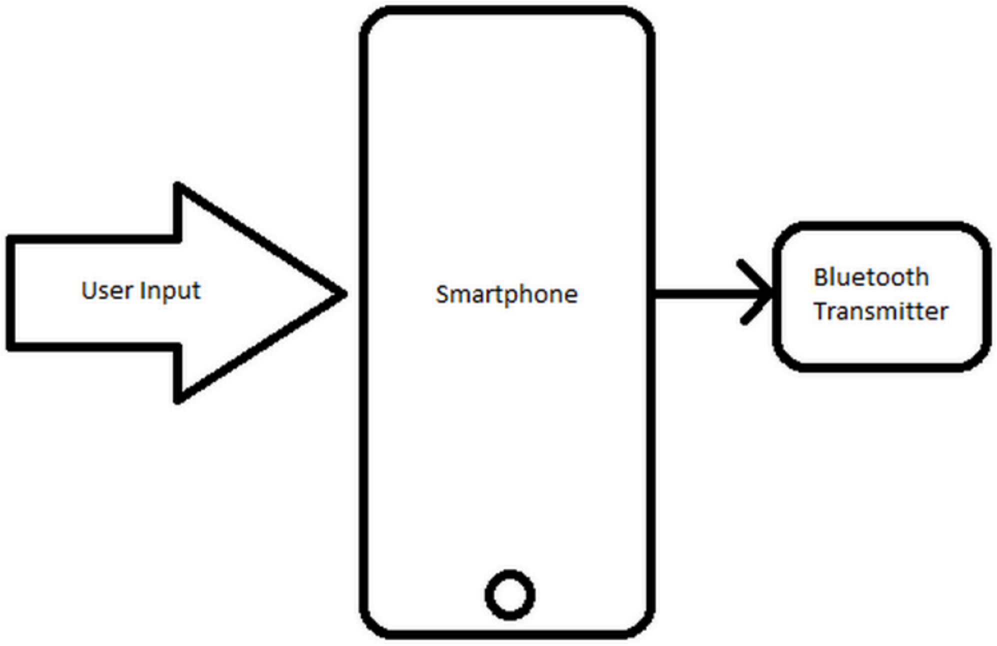


Figure 3: Block Diagram of Phone Control Unit

Control of the device is through the user's phone using a downloadable application. An overview of the user process is shown below.

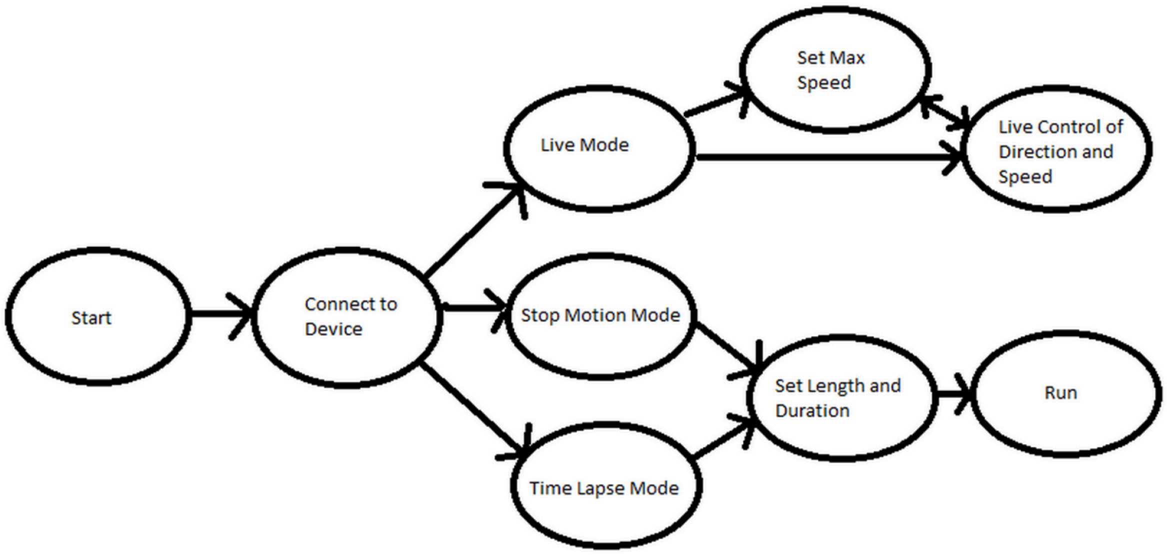


Figure 4: Block Diagram of User Control

As can be seen in the preceding diagram the user options are relatively limited. In the live mode they can control the slider in real time as well as setting maximum speeds. Stop motion and time-lapse are very similar to one another. The user sets the length and time desired and then the slider is set to move along the track. These two modes continue to function even when disconnect from the user app.

2.1 Use Case

There are three use cases for our product. The first, live mode is when a user wants to move their camera along a fixed rail to provide motion to their shot, while having the stability that a rail provides.

The other two use cases are time-lapse and stop-motion, which are used to provide sped up shots, with the difference being stop motion takes a sequence of pictures which are then stitched into a film, whereas time lapse takes a video which is subsequently sped up.

2.2 Top Level Design

The system design consists of three general sections:

- Hardware which provides the communication layer with the smartphone, as well as driving the motor which controls the motion of the camera
- Firmware which implements the control algorithm and Bluetooth communication between the hardware and the mobile application
- Mobile application software which interfaces with the user allowing control of the motor and mode selection for different operation of the device

3.0 Hardware Design

The hardware aspect of our project focuses on the mechanical systems used to drive the load on the camera slider, along with the electronic hardware used to communicate between the

3.1 Motor Unit

The motor unit was chosen to give smooth, reliable camera motion over a wide range of time and a variety of speeds. The unit itself, as shown in Figure 1999 is the SM-42BYG011-25, which is a 200 step, 1.8^o stepper motor. It is a perfect choice for our system, as the motor has an operating speed range exceeding what is required for any of our mode functionalities. Along with this, it is relatively small and lightweight, which factors into our mounting design. Finally, the motor is also common to find and fits well within our budget.

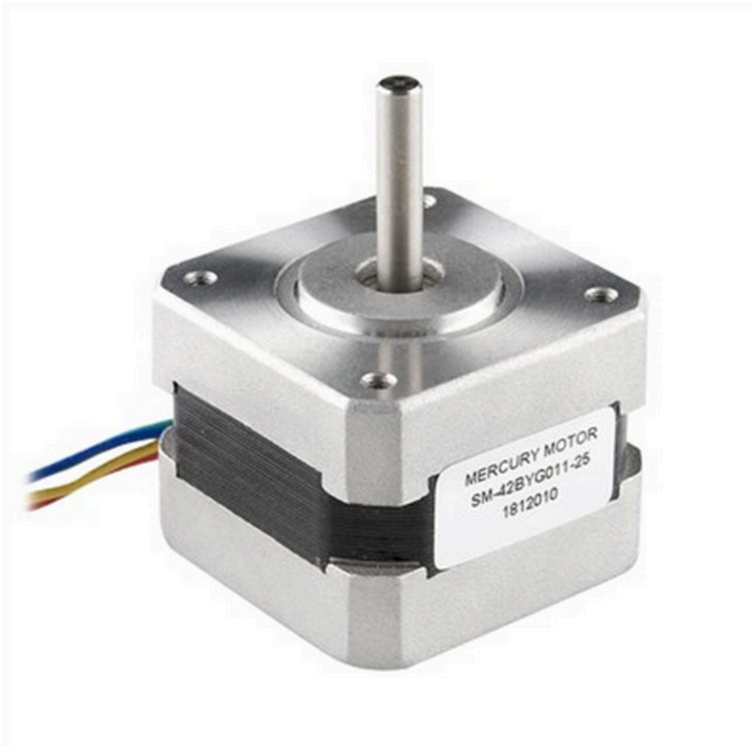


Figure 5: SM-42BYG011-25 Stepper Motor [8]

3.2 Microcontroller Unit

In order to control the physical motor system through wireless and electronic commands, we have chosen to use a popular and reliable microcontroller, the Arduino Uno, as shown in Figure 6. It offers a practical level of technology for the job at hand. Another consideration in using this product was the existence of open-source dedicated hardware built specifically to allow the microcontroller to generate Bluetooth connectivity.

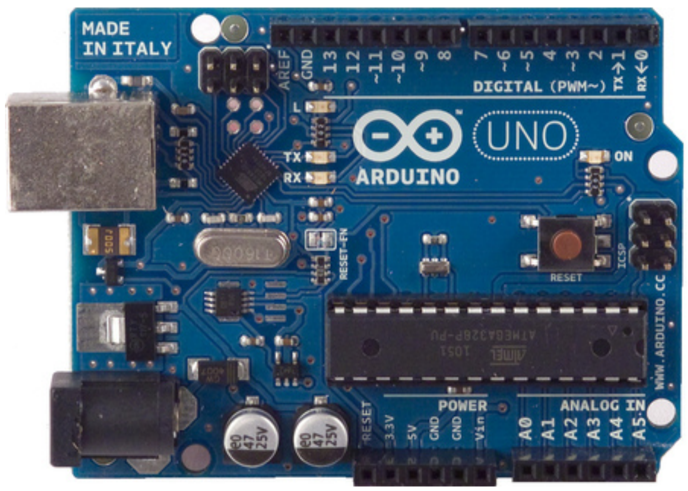


Figure 6: Arduino Uno Microcontroller [1].

However, due to some limitations occurring during our development phase - specifically that the Arduino has a low memory capacity - a duplicate system has been designed alongside, only using a Raspberry Pi. Benefits to using the Pi include giving more memory for the required functionality for the complete Hermes system, and a more natural programming environment for communication between the mobile app and computer. Some technical specifications comparing the Arduino Uno and the Raspberry Pi are shown in Table 1.

Specifications	**Arduino**	**Raspberry Pi**
Type of Board	Microcontroller	Computer
Clock Frequency	16 MHz	700 MHz
I/O Pins	14 digital, 6 analog	40 digital
Memory	32 KB (Flash), 2 KB (SRAM)	External (microSD)
Extra Features	Inexpensive, reliable, simplified programming environment	4 USB Ports, 3.5mm cable jack, HDMI port, Onboard GPU

Table 1: Comparison between Arduino Uno and the Raspberry Pi

Even though a short comparison would indicate the Pi being a more versatile resource for the job, we committed to the Arduino since we had previous experience with the architecture and it had a streamlined installation of the components that were required for the Hermes system. Not only this, but the Arduino Uno was the least expensive choice and the extra processing capabilities of the Pi would often be wasted (using extra power) considering the relatively simplistic actions needed for Bluetooth and motor control.

We would like to stress that at this stage in development, we are moving forward with testing for our current Arduino design plan, and that mentioning the Raspberry Pi is only for a precaution in regards to the worst case scenario in which the Pi will take the place of the Arduino. The rest of the document is focused on the construction and testing of our primary Arduino Uno setup.

3.3 Bluetooth System

As mentioned before, dedicated hardware was available for purchase that enabled Bluetooth capabilities to our microcontroller. The product itself is called the Adafruit Bluetooth LE nRF8001 Breakout and is shown in Figure 7. Justifications for choosing this product include open-source firmware libraries (as seen later in section 4.0) and it was well within our price range.

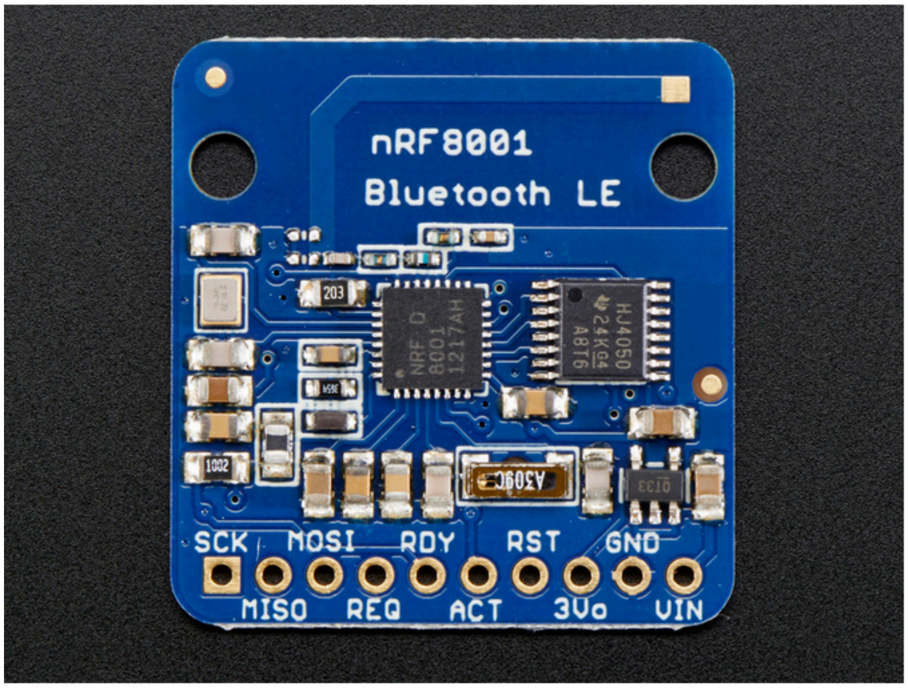


Figure 7: Adafruit Bluetooth LE nRF8001 Breakout [3]

Figure 8 shows the wiring diagram of the Bluetooth chip taken from the manufacturer's website.

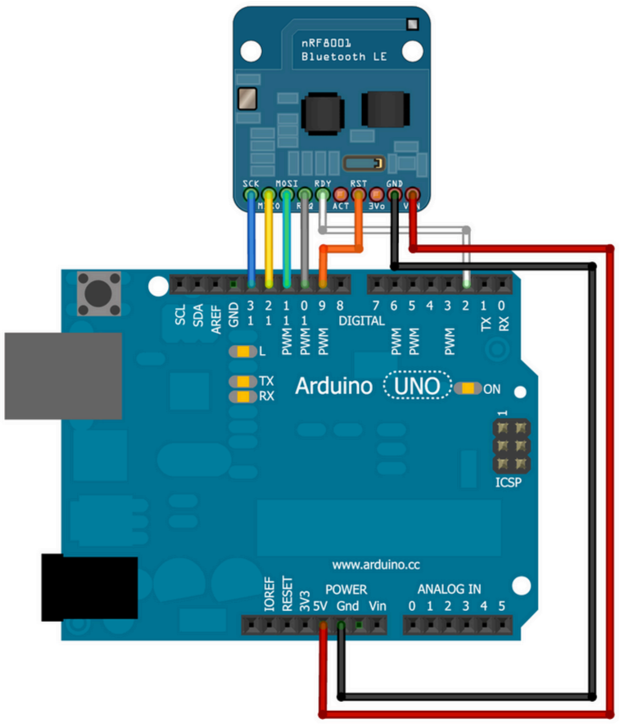


Figure 8: Adafruit Bluetooth LE nRF8001 Wiring Diagram [2]

3.4 Motor Shield

The motor shield is the interface between the microcontroller and the motor. The microcontroller on its own is not able to provide enough voltage and current through the output pins to power a motor, without risking damage to the chip. Our motor shield is the Adafruit Motor Shield V2 for Arduino, which can power up to 2 stepper motors or 4 DC motors as well as a servo. As we only require 1 stepper motor this board meets our requirements well. In addition it supports micro stepping [4] which meets [Req3.1.2.2 - PC] of the functional specifications. This provides the smooth motion required for slow movement without requiring a planetary gearbox.

The input voltage for the motor shield is 12 volts, which is provided through the Arduino.

The motor shield we used in our development is shown in Figure 9.

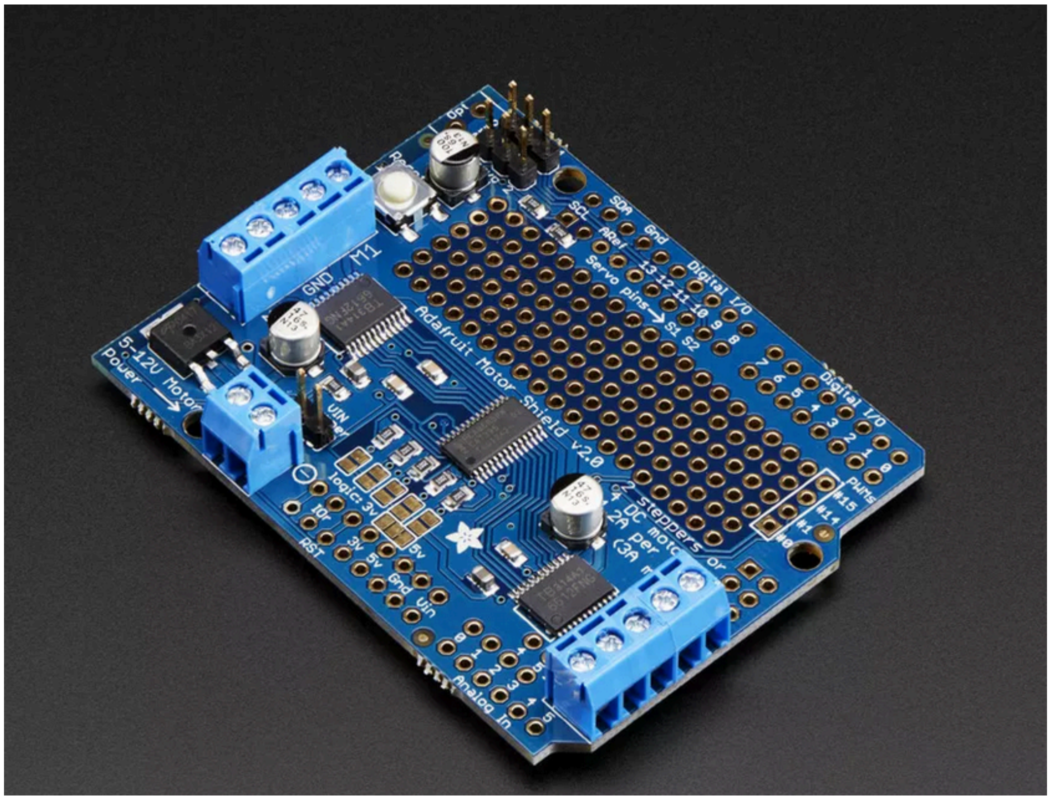


Figure 9: Adafruit Motor Shield V2 [4]

The motor shield is connected directly on top of the Arduino with a one-to-one connection with all the output pins of the Arduino. As such a wiring diagram is not included.

While the motor shield connects to all the pins on the Arduino it only uses 4-6 of them. It uses pins A4 and A5 for serial communication, either the 3V or 5V pin for power depending on the configuration, the ground as well as pins 9 and 10 if a servo is used [5]. All pins not accessed by the shield itself are passed through to be used by other shields, effectively acting as jumping cables.

Below is a wiring diagram, which demonstrates the connections between the motor shield and the motor. Either bank of 5 connectors may be used to connect the motor.

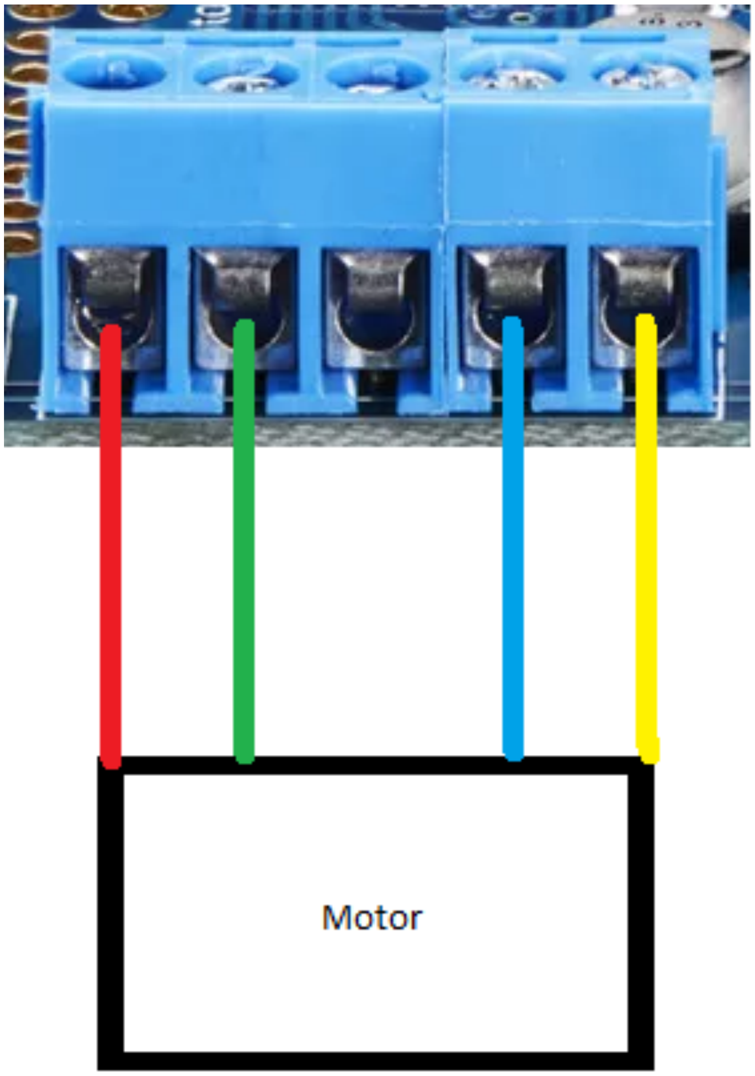


Figure 10: Wiring Diagram Between Motor and Motor Shield

3.5 Motor Mount

The mount is the part of the system, which acts to connect the driving force of the motor to the belt system of the Hedron camera slider. A diagram of the complete mounted component is depicted below.

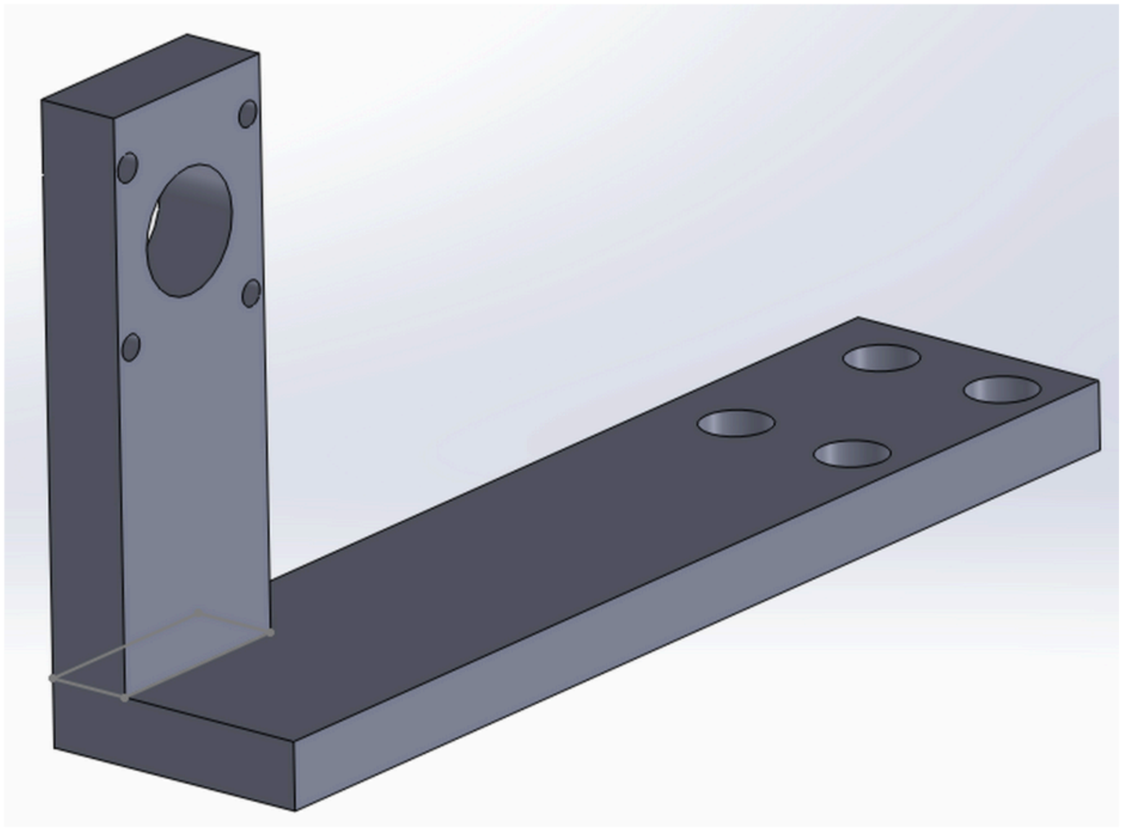


Figure 11: 3D Mount Diagram

3.5.1 Materials and Fabrication

The mount was fabricated using primarily a thin lightwood known as poplar board. This material made for easy construction as the material is soft enough so as to not damage the existing rail system, [Req3.0.3.1- PC], and to allow for clean cuts with a band saw, but still strong and rigid enough to support the weight of the motor. The softness of the material also allowed for easy drilling when constructing stepper motor fixation, a necessary feature when considering 4.5mm holes needed, a feature that would be extremely difficult to obtain with other materials such as sheet metal. Fabrication diagrams of the mounting system are included below.

3.5.2 Characteristics

The key characteristic of the mount that makes it ideal for our needs is the lightweight material. Given that in order to maintain the portability aspect, [Req3.0.1.8 - PD], of our functional specification, it is necessary to keep the overall weight of the product under 5 kg. Not only is the mount of negligible weight, but given that it replaces a bulkier steel component of the Hedron system, the total system weight is actually reduced. Additionally, by using wood as our primary material we follow the environmental requirements [Req3.0.2.2 - PT] of the functional specification.

3.6 Battery and Power Unit

In order to power the Arduino, as well as the stepper motor, a HitLights 12V Rechargeable Battery Pack (3800 mAh) is used due to its range of operation voltage falling well within the range needed by the Arduino/stepper motor combination [Req3.1.1.1 - PC]. The Bluetooth chip is low power and does not add significantly to the load.

4.0 Firmware Design

The firmware for the Hermes motion controller exists as the libraries and functions used in the compilation of the code used to program the various modules attached to the microcontroller. These libraries are essentially drivers for the extended hardware

4.1 Microcontroller

In order to program the Arduino Uno itself, we required a development environment. Fortunately, the Arduino team provides their own open-source IDE for software development across their various products. This environment contains the avr-gcc compiler, which itself is a subset of the regular GCC C++ compiler except designed specifically for microcontrollers that handle communication at the register level. Along with this, there are libraries available for simple projects that deal with LEDs or other basic circuit hardware. In addition, the Arduino IDE provides a serial console in order to allow enhanced debugging of the code running on the microcontroller.

4.2 Motor Shield

The motor shield that we acquired came with pre-existing libraries used to integrate the added motor functionality to the standard pins of the Arduino Uno. These libraries added support for various types of motors including DC motors, servomotors, and both unipolar and bipolar stepper motors. The latter of these is the one in which we will be making most use of.

Functions added to our inventory include those, which begin communication to the motor, set the frequency of the motor, set the current RPM and tell the motor to move. Successive uses of these last two functions are core in rotating the motor:

```
myMotor->setSpeed(RPM);
myMotor->step(STEPS, DIRECTION,MODE);
```

The names for the variables in this functions are quite descriptive. RPM and STEPS take unsigned integer values, DIRECTION allows forward and backward, and MODE is used for selecting specific operations of the stepper motor internals.

The modes taken by our bipolar stepper motor include SINGLE in which one coil of the motor is active, DOUBLE, in which two coils of the motor are active, INTERLEAVE, in which it alternates between single and double, and MICRO which adds micro step functionality. A better description on the details and testing of these modes is offered in Section 7.1.1.

4.3 Bluetooth

The Bluetooth chip procured from Adafruit came with pre-existing libraries used to access the functionality offered by the chip through the Arduino Uno microcontroller. Like the libraries mentioned before for the stepper motor, these were also imported into our IDE for use in our programming. Unlike the functions used for communicating from the smartphone application, the functions accessed on the microcontroller are mostly basic, and most of the action to produce the wireless communications is done on the nRF8001 chip. Some examples of functions used in the drivers for this device include setting up the pin outputs for the Arduino, polling the Bluetooth serial communication for information, and reading or writing across the wireless UART system.

5.0 Software Design

The Hermes Controller mobile app is the primary interface for user interaction and system feedback. Moreover, control via a wireless Bluetooth connection rather than a large, wired, and costly hardware controller uniquely positions the vitaMotu Hermes on the market [17]. Intuitive and hassle-free use of the product is therefore key to the functional and financial success of the Hermes.

The Apple iOS platform was chosen as the development platform for the Hermes Controller app during the proof-of-concept and prototype phases. While the production version of the Hermes will include both an iOS and Android compatible application, the iOS platform was chosen for initial development due to existing team expertise and iOS-only support offered by Adafruit, the Bluetooth chip manufacturer [18]. Future platform support will be determined through customer feedback and market penetration.

5.1 MVC Software Architecture

Following Apple's recommended guidelines, the Hermes Controller app is built around the Model-View-Controller (MVC) design paradigm [19]. MVC is designed to provide clear separation between the data (Model) the application is accessing and the UI (View). Bridging the gap between the data and the UI is the Controller [19].

UI driven actions (e.g. text input to be saved) trigger message calls from the UI to the controller. The controller performs any necessary logic or calculations and performs any necessary changes to the data model. Likewise, changes to the data model trigger controller notifications, which respond to the change by performing any necessary UI updates. Figure 12 illustrates the general MVC interactions. For more information regarding MVC design, see the Apple Developer Model-View-Controller guide [19].

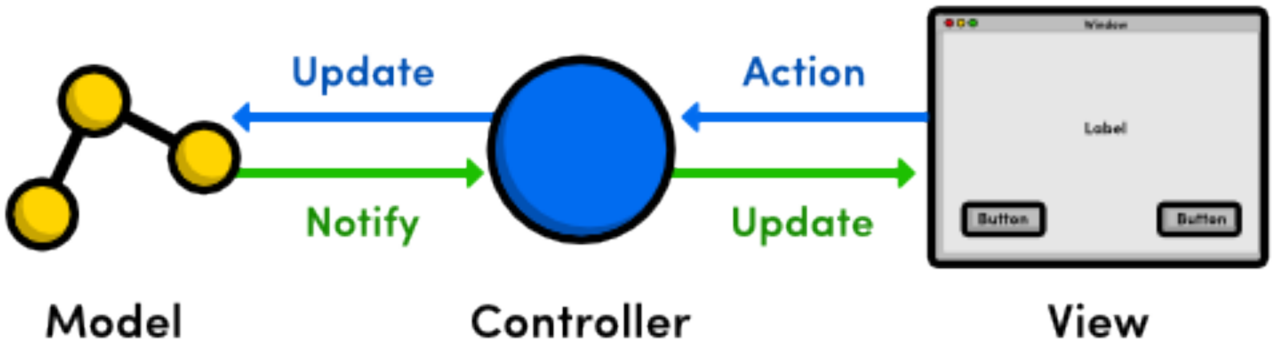


Figure 12: MVC Relationship Diagram

5.2 Data Models

The data model of the Hermes Controller app consist of two classes: BLEInterface and HermesControllerManager. These two classes facilitate all aspects of the Bluetooth discovery, connection establishment, and data transfer.

5.2.1 BLEInterface

The BLEInterface class is an adaption of the BLEAdapter library provided by Adafruit and acts as an intermediary between the HermesControllerManager and the CoreBluetooth framework provided by Apple [18]. This encapsulation has the benefit of isolating all CoreBluetooth framework calls within a single class and manages the Bluetooth connection at the lowest level. This includes initializing the Bluetooth hardware parameters, managing all stages of the connection, and performing the data transfer method calls. Only minor changes to this class were made from the original library, primarily to enhance clarity and readability for developers.

5.2.2 HermesControllerManager

The HermesControllerManager class is the forward facing class of the data model. Because the BLEInterface class is an adaption of a public library, it includes extraneous methods which complicate the interface with the controllers and does not provide necessary logic for application to the vitaMotu Hermes motion controller.

The HermesControllerManager is a singleton class which provides a single point of reference for any controller to establish a Bluetooth connection, determine the Bluetooth status, or perform data transfers. The public interface includes the following methods listed in Figure 13.

```
// Utility Methods
+ (HermesControllerManager *)sharedInstance;
- (void)scanForHermesController;
- (void)endScanForHermesController;
- (void)connectToHermesController:(CBPeripheral *)peripheral;
- (CBPeripheral *)getConnectedHermesController;
- (void)sendCommand:(NSString *)command;

// Operational Methods
- (void)beginRecording;
- (void)endRecording;
- (void)moveLeftWithSpeed:(NSInteger)speed;
- (void)moveRightWithSpeed:(NSInteger)speed;
- (void)beginTimeLapseWithDuration:(NSInteger)durationSeconds
                      startPosition:(NSInteger)start
                        endPosition:(NSInteger)end
                            damping:(NSInteger)damping
                               loop:(BOOL)loop;
- (void)beginStopMotionWithInterval:(NSInteger)intervalSeconds
                       startPosition:(NSInteger)start
                         endPosition:(NSInteger)end
                             damping:(NSInteger)damping;
```

Figure 13: Public Method Interface of HermesControllerManager

These methods facilitate an simple interface through which controllers in the app can easily interact with connected devices without the burden of connection management.

5.3 Controllers

Following Apple's recommended guidelines, apps contain numerous different controller classes [19]. At minimum, each View will have a dedicated Controller to interface with the data model, however additional controllers may exist to handle delegation, complex data structures, etc.

With the data model managing the low-level methods involved with the Bluetooth connection, the controllers in the Hermes Controller app are primarily responsible for general application logic. This includes responding to UI actions such as updating on screen labels and invoking public HermesControllerManager methods to control the Hermes motion controller (e.g. beginRecording).

While most of the operation of the Controllers is purely functional and beyond the scope of this document, the Bluetooth connection process and disconnection handling provided by the Controller warrants further discussion. For more information regarding the details of the Controller implementation see the public GitHub repository [21].

5.3.1 Bluetooth Connection Process

Upon launch of the Hermes Controller app, the Bluetooth scanning process begins immediately. Once a UART compatible peripheral has been detected, an alert appears on screen identifying the discovered device and requesting connection confirmation.

If confirmed by the user, a connection attempt is made and the user is informed of success or failure. If the user declines connection to the indicated device, the scan is continued and the cycle repeats as long as new devices are discovered.

If the 5 second scan timeout interval is triggered, the user is notified and, if any have been discovered, the user is presented with a list of all nearby peripherals available.

When a connection is made to a peripheral for the first time, the UUID is recorded for future use. Upon subsequent launches, the app first attempts to connect to any previously connected devices before beginning the process outlined above.

This process is designed to minimize time spent by the user during pairing. This process does not require the user to pair the device and peripheral prior to use and, because the of the use of a custom UART Bluetooth service, it is highly likely that the first device found is the intended peripheral. As a result, in most cases the only interaction required by the user is to confirm the connection, making the connection process virtually seamless for the user.

Figure 14 shows the flowchart for the Bluetooth connection process.

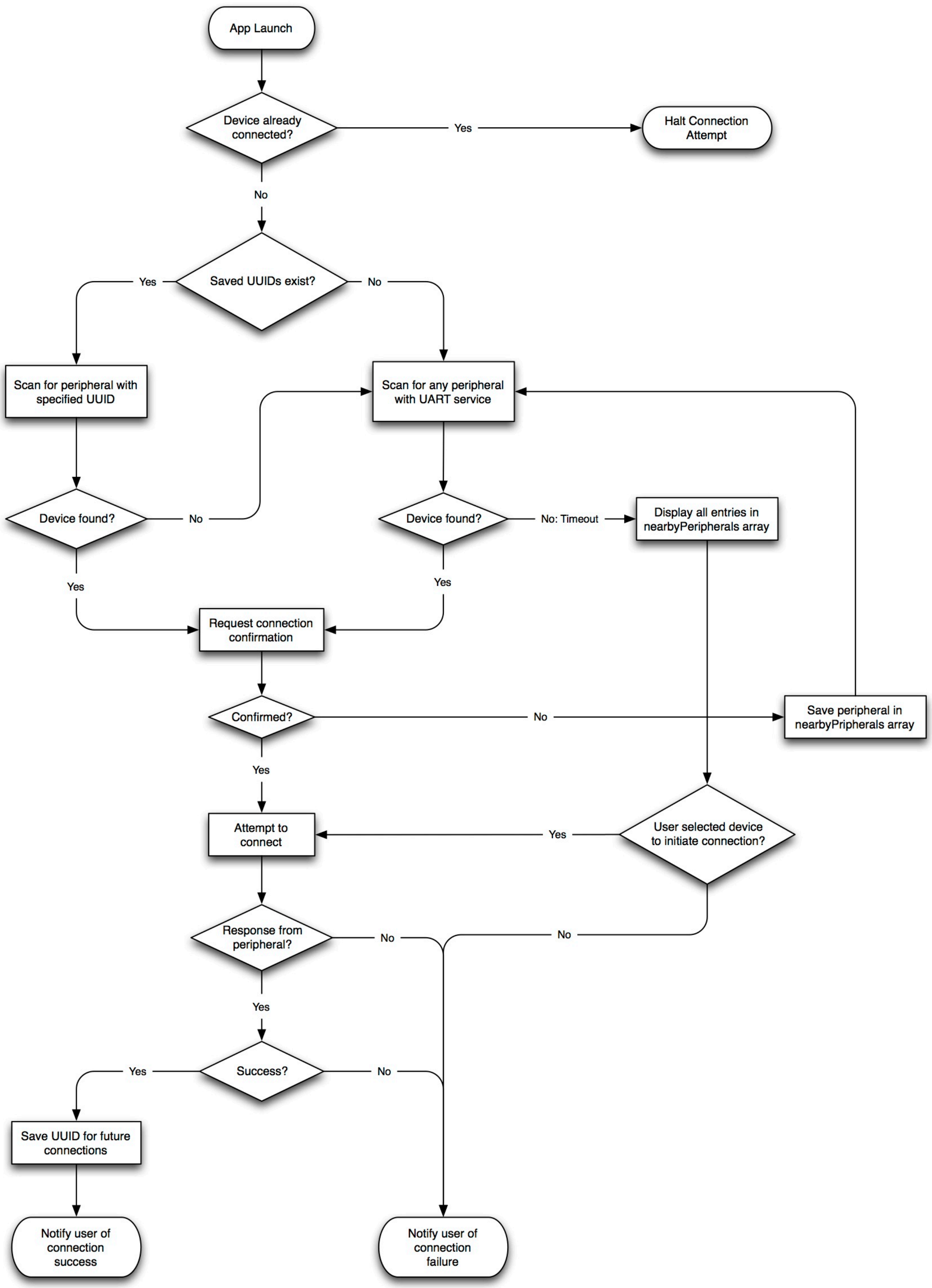

App Launch
Device already connected?
Yes
Halt Connection Attempt
No
Saved UUIDs exist?
Yes
No
Scan for peripheral with specified UUID
Scan for any peripheral with UART service
Device found?
No
Device found?
No: Timeout
Display all entries in nearbyPeripherals array
Yes
Yes
Request connection confirmation
Confirmed?
No
Save peripheral in nearbyPripherals array
Yes
Attempt to connect
Yes
User selected device to initiate connection?
Response from peripheral?
No
No
Yes
Success?
Yes
No
Save UUID for future connections
Notify user of connection success
Notify user of connection failure

Figure 14: Flowchart of Bluetooth Connection Process

5.3.2 Bluetooth Disconnection Handling

Upon Bluetooth disconnection a notification is sent from CoreBluetooth to the application. This notification triggers a UI alert notifying the user of the disconnection and given the option to initiate a new scan for peripherals.

5.4 Views

Because the Hermes Controller app is the primary means of user inaction with the motion controller, careful attention has been paid to the simplicity and ease of use of the mobile app. GUI design decisions have been made to follow the Apple Human Interface Guidelines as closely as possible.

The user interface is organized in a simple tab view, similar to the default Clock app provided in iOS. This enables the user direct access to the three modes of operation as well as the Help section at all times.

5.4.1 Live Mode

The Live Mode view is the simplest mode of operation and has only two adjustable parameters: Max Speed and Damping. These parameters are used to calculate the instantaneous speed to send to the controller when the Left and Right buttons are tapped. Lastly, a Record/Stop button controls camera-recording status. Figures 15 and 16 show the Live Mode view.

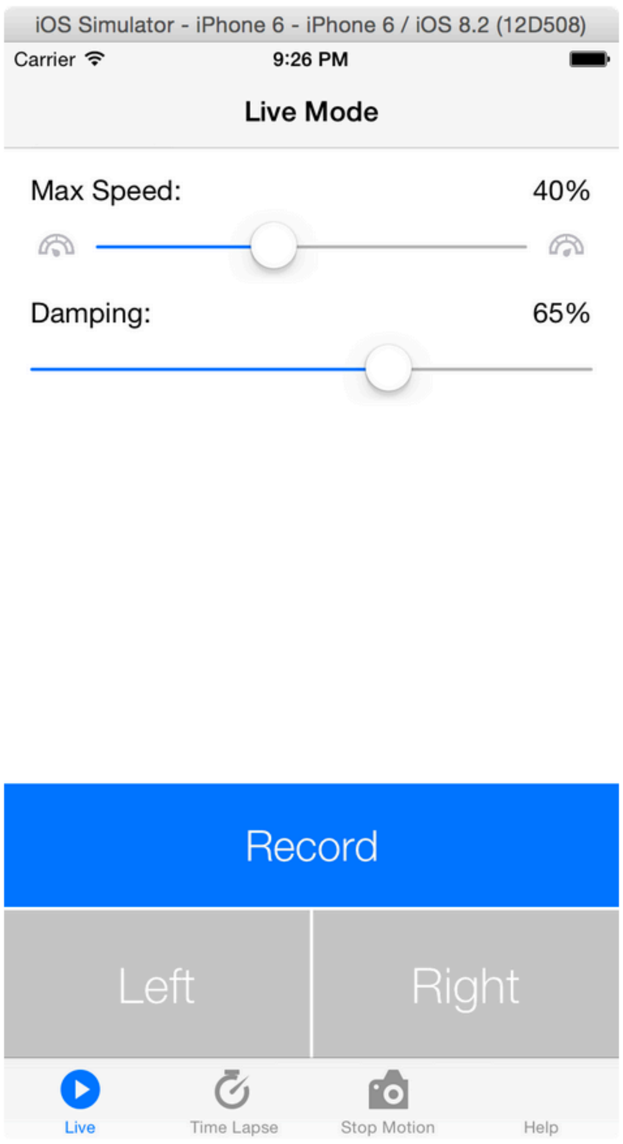


Figure 15: Live Mode GUI with Record Button

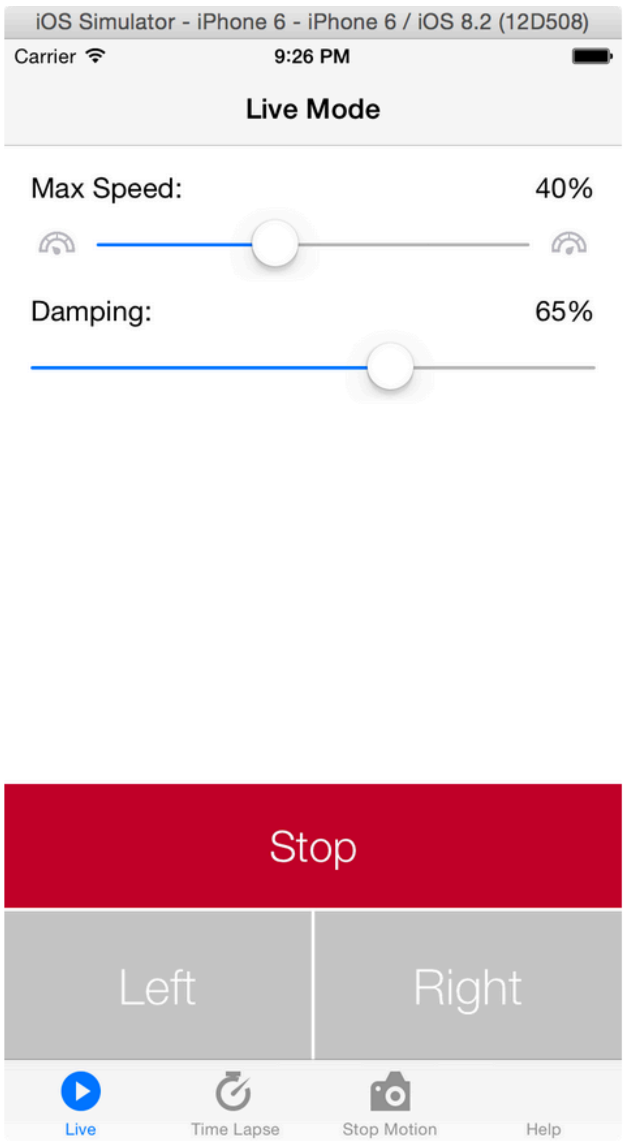
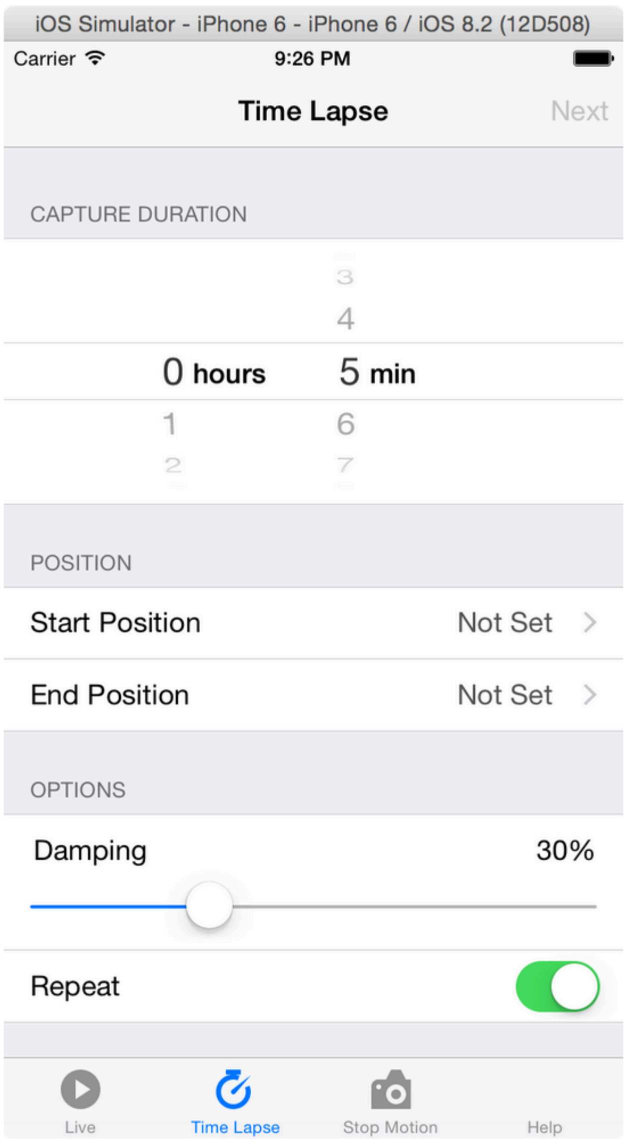


Figure 16: Live Mode GUI with Stop Button

5.4.2 Time Lapse Mode

The Time Lapse mode provides the ability to pre-program the motion controller movement. Users select the capture duration of the movement, the start and end position of the camera, the damping, and finally whether to repeat the motion upon completion. The main view of the Time lapse Mode is shown in Figure 17. Upon tapping the Start Position or End Position, a new view is loaded, shown in Figure 18, providing the user live control of the camera location on the track. Once the user has moved the camera to the desired location and tap Set the location is saved and the user returned to the view in Figure 17.



Time Lapse Next

CAPTURE DURATION

0 hours 5 min

POSITION

Start Position Not Set

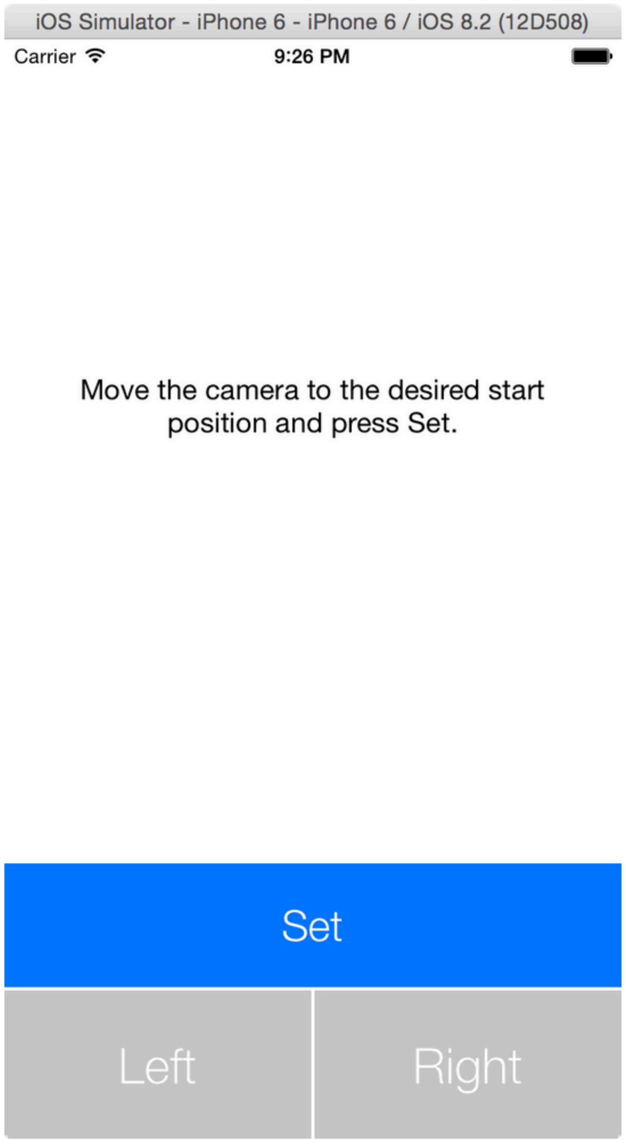
End Position Not Set

OPTIONS

Damping 30%

Repeat

Figure 17: Time Lapse Mode GUI

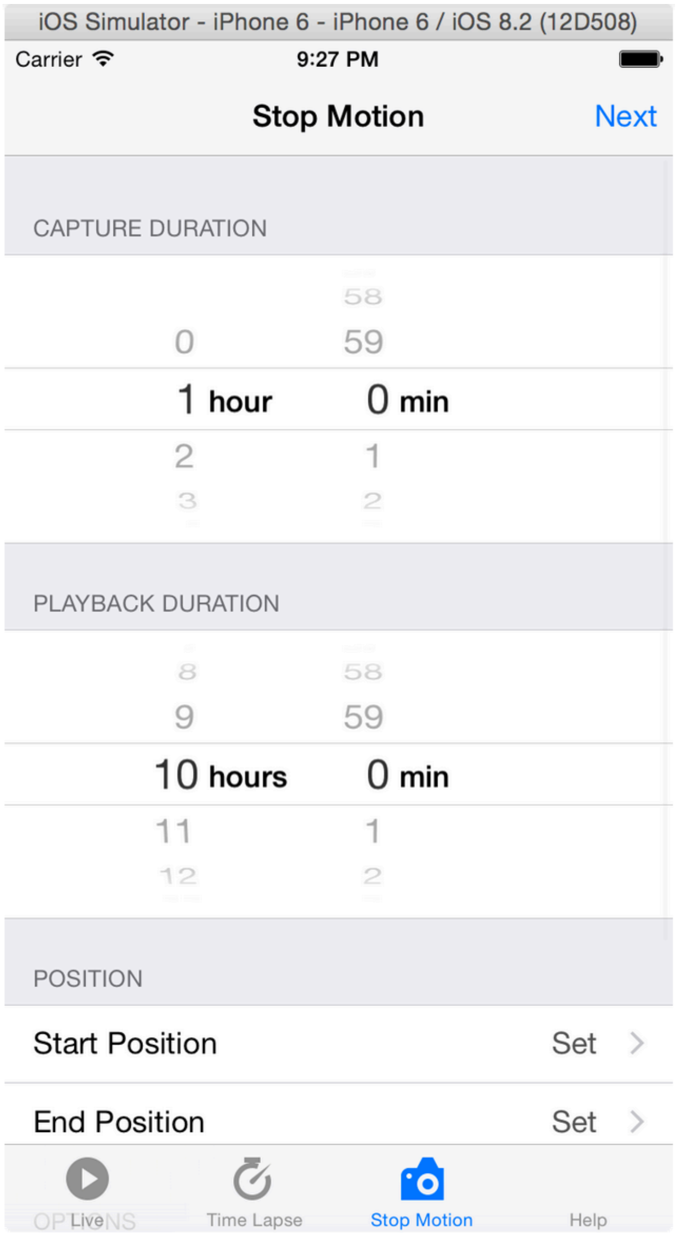


Move the camera to the desired start position and press Set.

Figure 18: Camera Position GUI

5.4.3 Stop Motion Mode

The Stop Motion Mode presents with an interface similar to the Stop Motion Mode allowing the user to specify the various parameters necessary for the stop motion movement. The Stop Motion view is shown in Figure 19. (Note: the units of the Playback Duration picker are mislabeled ‘hours’ and ‘min’ rather than ‘min’ and ‘sec’ as intended.)



CAPTURE DURATION

1 hour 0 min

PLAYBACK DURATION

10 hours 0 min

POSITION

Start Position Set

End Position Set

Figure 19: Stop Motion Mode GUI

5.4.4 Help

Lastly, a Help view is available to the user. This view contains a FAQ, contact information for question or bug reports, credits, and any necessary legal notices.

6.0 Bluetooth LE Connection

Bluetooth LE (also known as Bluetooth Low-Energy, Bluetooth 4.0, and Bluetooth Smart) is advancement to the wireless personal area network technology introduced as standard in 2010 [15]. The primary benefit of Bluetooth LE compared with prior versions is drastically reduced battery consumption [16]. Bluetooth LE chipsets can achieve a battery life of up to two years on a CR2302 (a common watch battery) [16].

While not all devices in use today support the standard, Bluetooth LE is backwards compatible with prior versions and Bluetooth SIG, the standardization body governing Bluetooth, predicts 90% support by 2018 [17]. Additionally, with competitively priced Bluetooth LE chipsets, the technology was clearly positioning as the most viable option for the vitaMotu Hermes [18].

6.1 Bluetooth LE Devices

Bluetooth LE compatible devices are classified as either Central or Peripheral devices depending on operation. Central and Peripheral Bluetooth devices follow a simple client-server paradigm where the Central device queries the Peripheral device for data over the connection at set intervals [19].

6.2 Connection

Connection over the Bluetooth LE protocol involves the exchange of 31 byte data packets between the Peripheral and Central devices known as Advertising Data and Scan Response Requests, respectively [19]. The exchange process is shown in Figure 20 below.

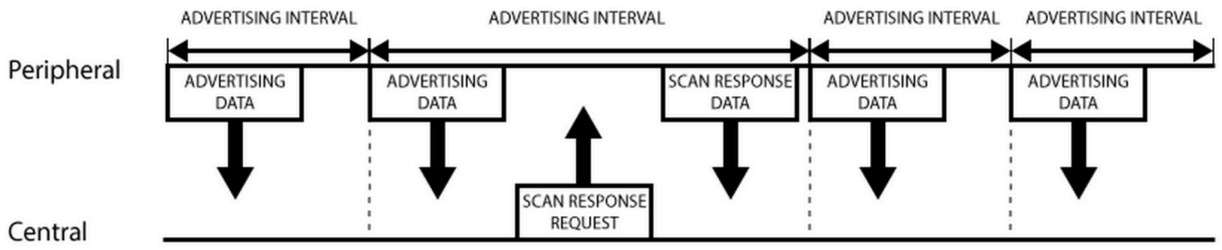


Figure 20: Exchange process [19].

Available Peripheral devices broadcast Advertising Data packets at to be intercepted by any Central device within range. Once broadcast, the peripheral device remains idle until a Scan Response Request is received or the advertising interval expires. This broadcast-idle cycle is fundamental to all stages of the Bluetooth LE connection/transfer and strongly contributes to the minimal energy consumption by compatible devices [19].

The Advertising Data packet contains Peripheral device information and is used, upon reception by a Central device, to initiate the connection. Connection is established once the Peripheral device receives the Scan Response Request from the Central device. Additionally, a Central device can request more information from the Peripheral prior to connection given in the form of a 31 byte Scan Response Data (see Figure 20). While a Central device can be connected to multiple Peripherals simultaneously, Peripheral do not support multiple connected devices and therefore stop broadcast of Advertising Data packets upon a successful connection [19].

6.3 Data Transfer

Data transfer via the Bluetooth LE specification makes use of Profiles, Services, and Characteristics organized as shown in Figure 21 below [20]. Profiles, Services, and Characteristics are conceptual groupings of related data used during the data transfer stage of a Bluetooth connection. Bluetooth SIG has defined a set of standard options for each, while third parties can create their own. In either case, a Profile, Service, or Characteristic is uniquely identified by either a 16-bit or 128-bit UUID for standard or custom offerings, respectively [20].

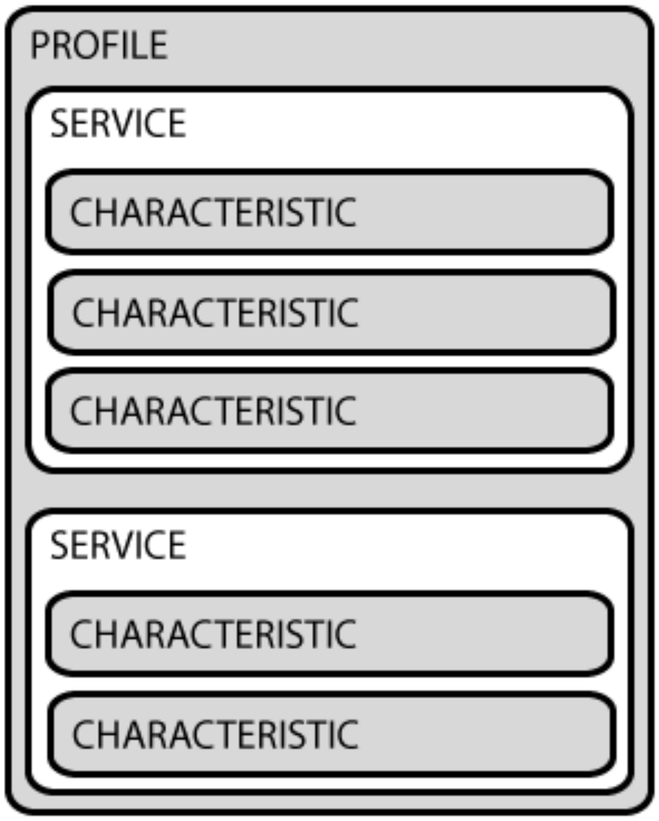


Figure 21: Hierarchy of Bluetooth LE Profiles, Services, and Characteristics [20]

Profiles simply represent a collection of services as defined by Bluetooth SIG or a peripheral designer. For example, the Heart Rate Profile is a profile defined by Bluetooth SIG, which includes the essential Services, and Characteristics necessary for a Bluetooth LE enabled heart-rate monitor [20].

Continuing down the hierarchy, Services are used to group collections of Characteristics (data points) into logical groupings of related data. Within the Heart Rate Profile, for example, exist the Heart Rate Service and Device Information Service, which are used to collect heart rate sensor data and heart rate sensor device information, respectively [20].

Lastly, Characteristics represent specific data points communicated between Peripheral and Central devices. Continuing with the example, the Heart Rate Service includes Characteristics for Heart Rate Measurement, Body Sensor Location and Heart Rate Control Point [20].

6.3.1 UART Implementation

Data transfer between the vitaMotu Hermes and the Hermes Controller app is accomplished via a UART-over-Bluetooth connection. As standard UART implementation is not provided by Bluetooth SIG, a custom UART communication channel was created by Adafruit for use with the nRF8001 Bluetooth LE board used in the Hermes motion controller [18]. The Service and Characteristics are outlined in Table 2.

Offering	UUID	Notes
UART Service	6E400001-B5A3-F393-E0A9-E50E24DCCA9E	UART Service which facilitates all data transfer between devices. The UUID is specified during scan and connection from a Central device to a compatible Peripheral
Tx Characteristic	6E400002-B5A3-F393-E0A9-E50E24DCCA9E	The transmit UART characteristic used for communicating from the Central device to a compatible Peripheral
Rx Characteristic	6E400003-B5A3-F393-E0A9-E50E24DCCA9E	The receive UART characteristic used for communicating from a compatible Peripheral to a Central device.

Table 2: Bluetooth LE Service and Characteristic Codes

6.4 Packet Communication

Once a connection has been established, data is transferred via 20 byte binary data packets between Central and Peripheral devices with the use of the `writeValue:forCharacteristic:type:` method provided by CoreBluetooth [21]

Note that while packets are transferred in the Peripheral discovery and connection, this exchange is handled internally by CoreBluetooth with the use of the `scanForPeripheralsWithServices:options:` and `connectPeripheral:options:` methods and does not make use of this custom packet format [21]. For a complete list of CoreBluetooth methods, see Apple CoreBluetooth Documentation for more info [21].

A custom packet format was devised to support all necessary communication between a mobile device and the Hermes Controller. In order to minimize the amount of data being transferred and the transfer latency, the packet format was specified as 20 bytes maximum to allow any command over a single packet. To ensure state consistency between the vitaMotu Hermes and the Hermes Controller app, an optional Peripheral response will be requested by the Central for all data transfers containing the current Peripheral state and parameters. This information is used to confirm operating mode, UI updating, and error handling/notifications.

6.4.1 Packet Transfer Intervals

The packet transfer frequency is determinant on the mode of operation. During Live Mode, slider movement must respond on demand to motion commands issued by the Hermes Controller app (Move Left, Move Right, etc.). As a result, constant communication between Central and Peripheral is required.

To facilitate minimal latency, the motion profile of the slider (i.e. velocity curve) is calculated on the mobile device based on the user specified Max Speed and Damping parameters. Once determined, the app sends simple Instantaneous Speed values to the device at ___ Hz. This enables the powerful CPU in the mobile device to handle calculations while the Arduino manages only basic motor control and Bluetooth communication.

Packets are assumed delivered to the connected device (a notification is sent from CoreBluetooth upon disconnect) so no feedback communication is sent from the vitaMotu Hermes to the Hermes Controller app in Live Mode. If a packet fails to be delivered to the motion controller, the slider will simply continue at the previously requested speed until another command is received. As commands are sent at regular intervals, a packet delivery failure does not provide significant interruption. In the event of a disconnection, camera movement ceases.

In the case of Time Lapse and Stop Motion modes, however, all parameters are fixed for the duration of the operation so the mobile device and Hermes unit do not require constant communication. As a result, a single packet is sent to the motion controller requesting the initiation of a Time Lapse or Stop Motion camera movement and the relevant parameters. The Arduino then performs the necessary motion profile calculations and executes the requested movement.

Once the initial command is sent, slider operation continues with or without a connected Central device. Throughout the movement, however, the Hermes motion controller broadcasts device status messages to the mobile device indicating the current progress or error status. The movement stops upon completion or receipt of a End Recording command from the Hermes Controller app.

6.4.2 Central to Peripheral Packet Transfer

The primary purpose of the connection is to allow commands to be sent from the Central device to a connected peripheral containing the user specified operating parameters. As such, the driving consideration while defining the packet format was the range of commands and parameters necessary to communicate. A complete list of operation commands sent from the mobile app to the vitaMotu Hermes is provided in Table 3.

Mode	Command	Parameters
Live Mode	Move Left	Instantaneous speed
Live Mode	Move Right	Instantaneous speed
Live Mode	Begin Recording	None
Live Mode	End Recording	None
Time Lapse Mode	Begin Recording	Duration Start Position End Position Damping Loop
Time Lapse Mode	End Recording	None
Stop Motion Mode	Begin Recording	Capture Interval Start Position End Position Damping
Stop Motion Mode	End Recording	None

Table 3: list of operation commands sent from the mobile app to the vitaMotu Hermes

The Modes and Commands in Table 3 were then assigned a binary representation. Next, each parameter was evaluated to determine the necessary range in values and allocated either 1 or 2 bytes and the appropriate unit representation of the parameter. Tables 4 and 5 below show the detailed breakdown.

Mode	Command	Byte Representation (Hex)
Live Mode	Move Left	00 00
Live Mode	Move Right	00 01
Live Mode	Begin Recording	00 02
Live Mode	End Recording	00 03
Time Lapse Mode	Begin Recording	01 00
Time Lapse Mode	End Recording	01 01
Stop Motion Mode	Begin Recording	02 00
Stop Motion Mode	End Recording	02 01

Table 4: List of Command Packet Suffices

Parameter	Packet Order	Byte Length	Byte Representation (Hex)
Live Mode			
Instantaneous speed	1	1	Value in range 0-100 representing percent of maximum speed
Time Lapse Mode			
Duration	1	2	Value in range 0-65,535 representing the duration of the time lapse recording in seconds
Start Position	2	2	Value in range 0-65,535 representing the start position of the camera movement in motor steps

End Position	3	2	Value in range 0-65,535 representing the start position of the camera movement in motor steps
Damping	4	1	Value in range 0-100 representing percent of maximum damping
Loop	5	1	Binary on/off value
Stop Motion Mode			
Capture Interval	1	2	Value in range 0-65,535 representing the duration between camera exposures in seconds
Start Position	2	2	Value in range 0-65,535 representing the start position of the camera movement in motor steps
End Position	3	2	Value in range 0-65,535 representing the start position of the camera movement in motor steps
Damping	4	1	Value in range 0-100 representing percent of maximum damping

Table 5: Detailed Breakdown of Packet Parameters

Tables 4 and 5 provide all specifications necessary to create a packet for transfer. Note that while the packet format varies between modes, the first byte of the packet always specifies the mode of operation and therefore also the packet format. This information is used by the Hermes' on-board Arduino micro controller to parse the packet structure and parameters.

Example packets in Tables 6 to 7 illustrate the packet structure outlined above.

Data	00	01	23	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Byte	1	2	4	4-20
Command				Live Mode - Move Right at 35% max speed

Table 6: Example Live Mode Packet

Data	00	03	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Byte	1	2	3	4-20
Command				Live Mode - End Recording

Table 7: Example Live Mode Packet

Data	01	00	0E 10	00 64	0B B8	4B	00	00 00 00 00 00 00 00 00 00 00
Byte	1	2	3-4	5-6	7-8	9	10	11-20
Command								Time LApse Mode - Start Recording for 3600 seconds between step 100 and step 3000 with 75% damping and no looping

Table 8: Example Time Lapse Mode Packet

Data	02	00	00 78	00 96	09 C4	32	00 00 00 00 00 00 00 00 00 00
Byte	1	2	3-4	5-6	7-8	9	11-20
Command							Stop Motion Mode - Capture a photo every 120 seconds while moving between step 150 and step 2500 with 50% damping

Table 9: Example Live Mode Packet

Note that all unused bits or parameters not in use are padded with zeros.

6.4.3 Peripheral to Central Packet Transfer

Status messages are sent from the Hermes motion controller to the Hermes Controller app while operating in Time Lapse or Stop Motion modes. These messages indicate the current percent progress of the movement and if any errors have occurred. These packets have a simple form with the first byte indicating if an error occurred and the second byte representing either the progress (in percent) or the error code. Examples of each case are shown in Tables 10 and 11.

Data	00	57	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Byte	1	2	3-20
Command			Device OK - 87% complete

Table 10: Example Status OK Packet

Data	00	65	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Byte	1	2	3-20
Command			Device Error - Error 101 occurred

Table 11: Example Status Error Packet

7.0 Enclosure Design

The enclosure is designed to be sleek and weather resistant. This will be achieved by using recyclable aluminum [3.0.2.2 - PT] with tight seals, meeting [3.0.1.2 - PT] and [3.0.1.3 - PD].

For the proof of concept stage the enclosure is a wooden box of dimension 5cm x 10cm x 20cm. It will be composed of poplar board and fastened together using screws. Holes will be cut out to for switches and indicator lights, as well as a LANC connection and a charging port.

The enclosure houses all the components besides the charger and smartphone. It covers the motor, ensuring user safety [3.0.3.5 - PD]. As well as the rounded corners protect the user from unintentional cuts [3.0.3.3 - PD]. The small size meets requirement [3.0.1.8 - PD] allowing the user to maintain the relative portability of their slider. A mockup of the front of the device is shown below.

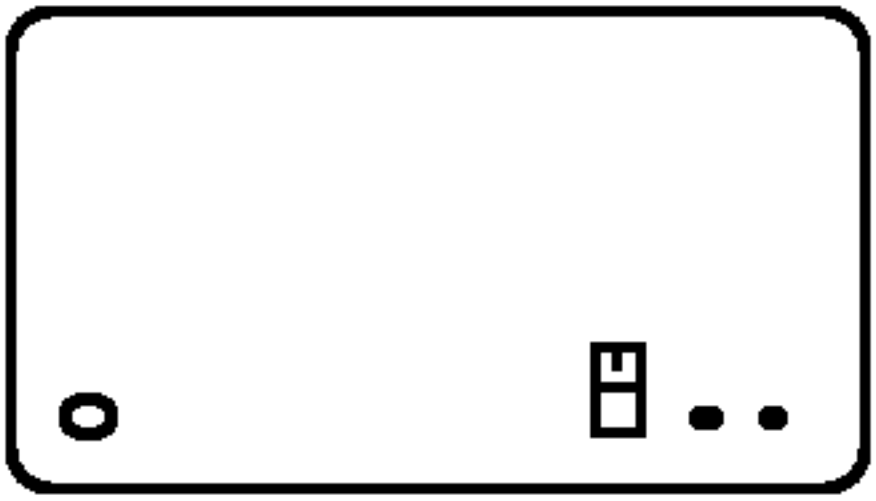


Figure 22: Mockup of Enclosure front

8.0 Test Plan

Due to the fact that our system is aimed towards a semi-professional audience, the quality and usability of each component is top priority. In order to ensure the highest level of performance from the Hermes system, unit testing will be conducted in order to isolate ideal functionality of each system as they are completed, as well as a form of regression testing to insure continued functionality of every system as integration takes place. In this way, after the regression phase of testing, the system as a whole should work without interference from previously tested models.

8.1 Unit Testing

This first wave of testing will be performed on the individual components of the system as follows:

8.1.1 Stepper Motor

Speed functionality - Set motor to min and max speeds respectively and counted turns of the shaft to ensure set RPM values matched the measured RPM.

Directional Functionality - Switch motor settings between forward and reverse direction and observe change in functionality.

Mode Functionality - Switch between Single, Double, Intermittent, and micro step modes and observe the differences in motion. Single should show the motor rotating at the desired RPM but with less torque than what is rated. Double should have the motor rotating at the desired RPM only with considerably more torque than what is found with single stepping. Intermittent should have jerky movement while rotating (and is largely unused in our project). Micro stepping should make extremely small steps and is tested with lower RPM values, as it will be implemented for the longer time varying shots.

8.1.2 Microcontroller

In order to test the microcontroller a number of small projects will be first be run to full explore its capabilities. Additionally, the microcontroller will be integrated with the stepper motor, the Bluetooth chip, and the motor shield in order to insure programming is done correctly and also to insure the MCU is capability of the high level of functionality demanded by the system.

8.1.3 Bluetooth Test Plan

In order to test the Bluetooth chip, and Bluetooth test app was used in order to send basic commands from the cell phone iOS to the microcontroller as required by [Req3.2.3.1 - PC]. These commands include sending of basic text, as well as some basic real time motor control.

8.1.4 Smartphone App

The primary function of the Hermes Controller mobile app can be broken down into two general categories: Bluetooth connection and UI.

8.1.4.1 Bluetooth Software

The Bluetooth software test plan requires the application to be able to successfully accomplish the following:

- Discover nearby peripherals advertising the custom UART service
- Connect to a specified peripheral device
- Transmit command packets to the peripheral device
- Receive status packets from the peripheral device

8.1.4.2 UI

Testing of the Hermes Controller app involves ensuring the following can be accomplished without noticeable bugs or crashing:

- Set all Live Mode parameters with on screen feedback and correctly respond to all button interactions
- Set all Time Lapse Mode parameters with on screen feedback and correctly respond to all button interactions
- Set all Stop Motion Mode parameters with on screen feedback and correctly respond to all button interactions
- Provide Help view with relevant information

8.1.5 Motor Control Test Plan

The motor control is tested by ensuring the three following section function as expected.

8.1.5.1 Stop Motion

For stop motion a distance and time will be inputted into the device. Then, the device will be filmed to ensure it moves over the proper period of time and then the distance will be measured to ensure correctness.

In phase two of this testing, once functional LANC protocol has been implemented a camera will be placed on the slider and it will be checked to see whether the LANC connection is synced properly by looking at the recorded video. If there are blurry frames, this indicates a malfunction of the timing. If not, there is no problem.

8.1.5.2 Time Lapse

Time lapse is identical to stop motion testing except phase two is not necessary because the controller does not affect the functioning of the camera; real-time video is taken instead of pictures every interval.

8.1.5.3 Real Time

Real time motion is tested by inputting various maximum speeds and checking, whether they are obtained by measuring the distance and time taken in rough calculations.

8.2.1 Regression Testing Phase 1

Regression testing will first be performed upon completion of systems outlined as a necessity of proof of concept functionality (section 3.0 of the functional specification). These systems include fluid stepper motor motion, minimum speed of 0.06 RPM [Req3.1.2.5 - PC], maximum speed of 215 RPM [Req3.1.2.3 - PC], visible status lights, stepper motor control, Bluetooth connectivity, and basic app functionality.

After the functionality of the basic systems is confirmed the systems will then be integrated in order to insure that data transfer between them functions as expected. In order to make the debugging of this system manageable systems will be integrated one at a time.

We will first connect the motor to the MCU and test that the motor output corresponds to the inputs sent by the MCU. This will be accomplished by performing the motor test outlined above with inputs sent from the Arduino. The Bluetooth chip well then be added, and in conjunction with the test app, motor control testing will be performed again with commands from the Bluetooth. The smart phone app will then be integrated and the basic motor controls will be performed again using the GUI of the smartphone app. Finally advanced motor motion, such as the stop motion, time lapse, and real time motion will be tested using commands from the smartphone.

8.2.2 Regression Testing Phase 2

A second phase of regression testing will ideally be performed following the addition of extra features not essential to the construction of a functioning prototype. Features that we wish to add include interaction with a camera via LANC protocol, power and connectivity buttons, and make system highly transportable. In a similar fashion to be previous regression test section, the additional features will be integrated sequentially.

During this section of testing the LANC protocol camera communication will be used in order to allow pictures to be taken at set intervals during the stop motion mode for the requirements outlined in [Req3.2.2.5 - PT]. This will be tested by simply trying to take pictures through commands from the MCU. Next the power and connectivity buttons will be added and integrated with the MCU and tested by simply pressing the buttons and checking for desired effect. Lastly, most difficult to test is the portability of the device. In order to do this the device must be packed up and moved and then re-set up and test all facets of functionality repeatedly in order to insure device is sufficiently durable.

9.0 Conclusions

This document details the design of our Hermes camera motion controller. Combined with the functional specifications document it provides and overview of the design and functioning of our product. This design corresponds to three major components:

- The hardware design which provides the mechanical means and electronic circuits required for the movement of the slider.
- The firmware design which controls the various circuits and motor to achieve the smooth and steady motion of the product.
- The software design which provides a means for the users to input their desired movement patterns.

These components have all been detailed in various sections in the above document, which justifications provided to prove our design.

References

[1] Arduino.cc, 'Arduino - ArduinoBoardUno', 2015. [Online]. Available: http://www.arduino.cc/en/Main/arduinoBoardUno. [Accessed: 20- Mar- 2015].

[2] Learn.adafruit.com, 'Hooking Everything Up | Getting Started with the nRF8001 Bluefruit LE Breakout | Adafruit Learning System', 2015. [Online]. Available: https://learn.adafruit.com/getting-started-with-the-nrf8001-bluefruit-le-breakout/hooking-everything-up. [Accessed: 18- Mar- 2015].

[3] T. others, 'Bluefruit LE - Bluetooth Low Energy (BLE 4.0) - nRF8001 Breakout [v1.0] ID: 1697 - $19.95 : Adafruit Industries, Unique & fun DIY electronics and kits', Adafruit.com, 2015. [Online]. Available: http://www.adafruit.com/product/1697. [Accessed: 18- Mar- 2015].

[4] T. others, 'Adafruit Motor/Stepper/Servo Shield for Arduino v2 Kit [v2.3] ID: 1438 - $19.95 : Adafruit Industries, Unique & fun DIY electronics and kits', Adafruit.com, 2015. [Online]. Available: http://www.adafruit.com/products/1438. [Accessed: 18- Mar- 2015].

[5] Learn.adafruit.com, 'FAQ | Adafruit Motor Shield V2 for Arduino | Adafruit Learning System', 2015. [Online]. Available: https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/faq. [Accessed: 18- Mar- 2015].

[6] Aislelabs, 'The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs - Aislelabs', 2015. [Online]. Available: http://www.aislelabs.com/reports/beacon-guide/. [Accessed: 20- Mar- 2015].

[7] Cinevate.com, 'Hedron Camera Slider', 2015. [Online]. Available: http://www.cinevate.com/store2/catalog/product/gallery/id/244/image/729/!. [Accessed: 20- Mar- 2015].

[8] S. Cable, A. R3, E. Driver, B. Straight, A. 5V/16MHz, R. total), S. steps/rev) and U. Aluminum, 'Stepper Motor with Cable - ROB-09238 - SparkFun Electronics', Sparkfun.com, 2015. [Online]. Available: https://www.sparkfun.com/products/9238. [Accessed: 18- Mar- 2015].

[9]B. Edalat, B. Keane, M. Palibroda, J. Raine and C. Scott, 'Project Proposal, ENSC 305W/440W Capstone Project', 2015.

[10] adafruit.com, 'Adding App Support | Getting Started with the nRF8001 Bluefruit LE Breakout | Adafruit Learning System', 2015. [Online]. Available: https://learn.adafruit.com/getting-started-with-the-nrf8001-bluefruit-le-breakout/adding-app-support. [Accessed: 20- Mar- 2015].

[11] Developer.apple.com, 'Model-View-Controller', 2015. [Online]. Available: https://developer.apple.com/library/mac/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html. [Accessed: 20- Mar- 2015].

[12] Rypress.com, 'The MVC Pattern - Ry's Cocoa Tutorial - RyPress', 2015. [Online]. Available: http://rypress.com/tutorials/cocoa/the-mvc-pattern. [Accessed: 20- Mar- 2015].

[13] GitHub, 'justinraine/HermesControllerApp', 2015. [Online]. Available: https://github.com/justinraine/HermesControllerApp. [Accessed: 20- Mar- 2015].

[14] Developer.apple.com, 'iOS Human Interface Guidelines: Designing for iOS', 2015. [Online]. Available: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/. [Accessed: 20- Mar- 2015].

[15] Slashgear.com, 'Bluetooth 4.0 finalized: low power mode & boosted range - SlashGear', 2015. [Online]. Available: http://www.slashgear.com/bluetooth-4-0-finalized-low-power-mode-boosted-range-2182619/. [Accessed: 20- Mar- 2015].

[16] Aislelabs, 'The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs - Aislelabs', 2015. [Online]. Available: http://www.aislelabs.com/reports/beacon-guide/. [Accessed: 20- Mar- 2015].

[17] Bluetooth.com, 'Mobile Telephony Market| Bluetooth Technology Website', 2015. [Online]. Available: http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx. [Accessed: 20- Mar- 2015].

[18]T. others, 'Bluefruit LE - Bluetooth Low Energy (BLE 4.0) - nRF8001 Breakout [v1.0] ID: 1697 - $19.95 : Adafruit Industries, Unique & fun DIY electronics and kits', Adafruit.com, 2015. [Online]. Available: http://www.adafruit.com/product/1697. [Accessed: 20- Mar- 2015].

[19] Learn.adafruit.com, 'GAP | Introduction to Bluetooth Low Energy | Adafruit Learning System', 2015. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap. [Accessed: 20- Mar- 2015].

[20] Learn.adafruit.com, 'GATT | Introduction to Bluetooth Low Energy | Adafruit Learning System', 2015. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt. [Accessed: 20- Mar- 2015].

[21] Developer.apple.com, 'About Core Bluetooth', 2015. [Online]. Available: https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html. [Accessed: 20- Mar- 2015].