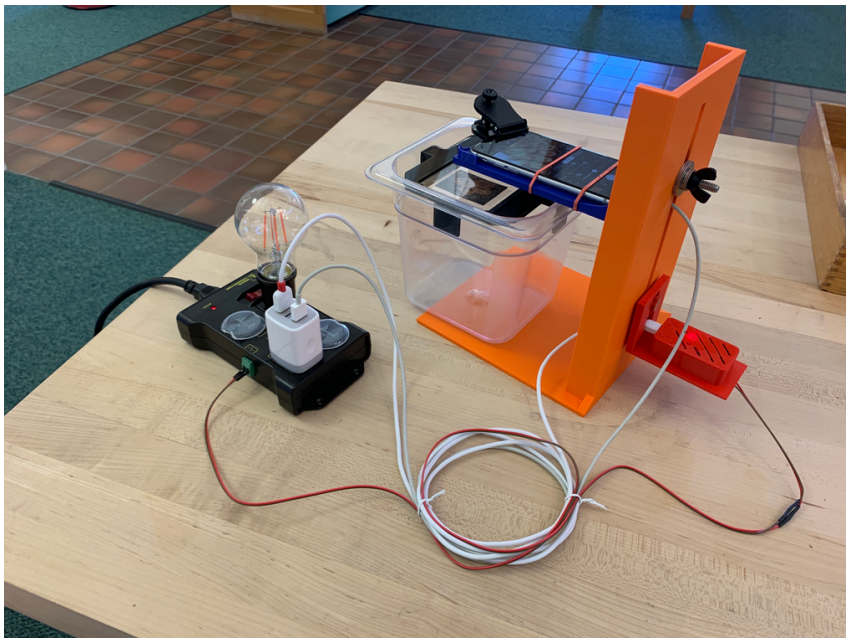
KISS Smartphone Time-Lapse Controller

2023-04-26 Rev 0.9

The KISS Controller is used to create time-lapse videos or to take scheduled video recordings from a single position using a Smartphone Camera. The KISS Controller is operated by a web based user interface running on a Smartphone or Laptop to direct the sequence of the video recording or photographs to create a time lapse video.

The components of the KISS system include an adjustable height camera holder, an ESP32 WROOM microcontroller which provides a bluetooth keyboard for the camera shutter to take photos and a WiFi access point for the User Interface, and a controllable four outlet power relay module that the microcontroller uses to turn on a light during the video recording or photography. In addition, the system includes a Python program to watermark the photos and create the time-lapse video.

The purpose of this document is to describe the setup, operation, DIY construction, hardware, and software of the KISS Smartphone Time-Lapse Controller.



Tom Rolander, MSEE
Mentor for Circuit Design, Software, 3D Printing
Miller Library, Fabrication Lab
Hopkins Marine Station, Stanford University,
120 Ocean View Blvd, Pacific Grove, CA 93950
+1 831.915.9526 | rolander@stanford.edu

Table of Contents

KISS Controller Setup 4
- iPhone 4
  - Camera Setup 4
    - Bluetooth for Camera Shutter 4
    - Camera Always ON 7
  - User Interface 10
    - KISS WiFi Access Point 10
- Android 14
  - Camera Setup 14
    - Bluetooth for Camera Shutter 14
    - Android Always ON 18
    - Android Allow Data Roaming 23
  - Android User Interface 28
    - KISS WiFi Access Point 28

KISS Controller Operation 32
- Settings 32
  - Number of Loops 34
  - Video Record Seconds 34
  - Light Delay Before Seconds 34
  - Light Delay After Seconds 34
  - CameraID 34
- Photo Time Lapse and Video Time Lapse 35

KISS Controller Construction 37
- 3D Printed Parts 37
  - Microcontroller Enclosure and Bracket 37
  - Smartphone Holder and Bracket 38
  - Stand for Microcontroller and Smartphone Brackets 39
- Parts List 40
  - ESP32-WROOM-32 Microcontroller 40
  - Controllable Four Output Power Relay Module 41

Wires 41
KISS Controller Assembly 42
Arduino (ESP32 WROOM) Source Code for KISS 43
KISS Image Processing and Video Creation 66
Installation and Setup for Video Creation 66
Install Python 66
Install Image Magick 66
Install ffmpeg 66
Detailed Mac Instructions 66
Install Homebrew 66
Install Python 66
Install Image Magick 66
Install ffmpeg 66
Install pip 66
Install exifread 66
Install scipy 67
Install VideoCreation Python Program 67
Create folder 67
Download VideoCreation.py 67
Use TextEdit to save VideoCreation.py 67
Verify execution of VideoCreation.py 67
VideoCreation 68
VideoCreation Python Program Operation 68
VideoCreation Python Source Code 69

KISS Controller Setup

The KISS Controller is operated with commands entered on a Smartphone User Interface. The KISS Smartphone Camera uses a Bluetooth device to receive shutter commands. The following steps describe the setup for both the iPhone and Android Smartphones that you use as a camera. You can use any combination of iPhones and Androids.

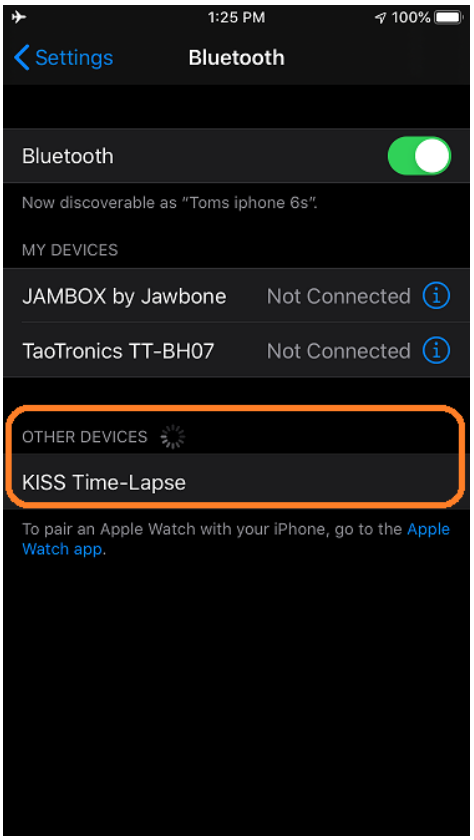
Before you connect either an iPhone or an Android to be used as a camera for the KISS Controller you must turn on the power for the KISS Microcontroller by connecting a cable to plug it into a USB adapter.

iPhone

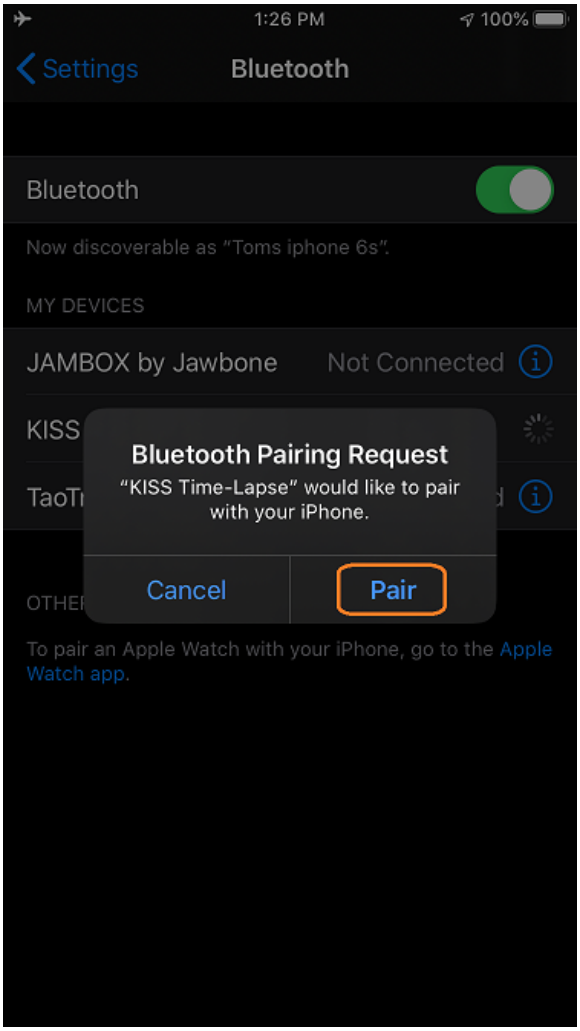
Camera Setup

Bluetooth for Camera Shutter

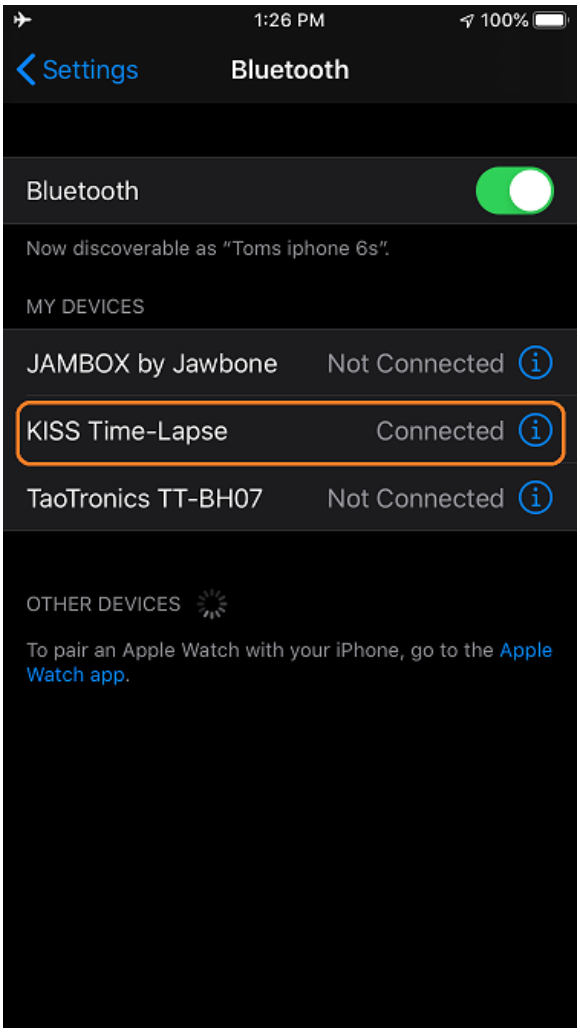
1. Go to Settings -> Bluetooth
   Under OTHER DEVICES you should see KISS Time-Lapse.



2. Select the KISS Time-Lapse and you will be prompted with a pairing request from the KISS Microcontroller. Click on the “Pair” button.

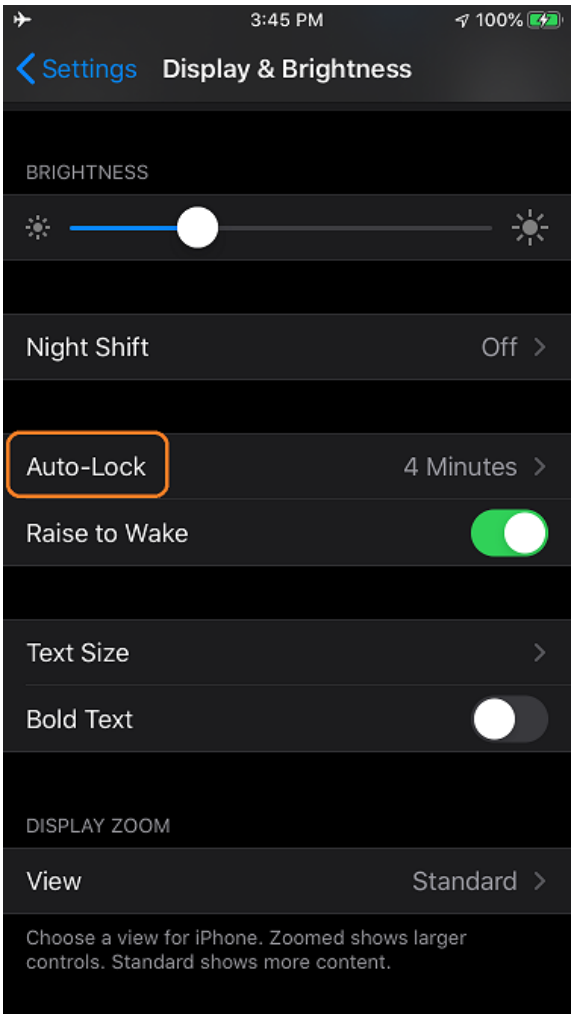


3. When the pairing is completed you will see that the KISS Time-Lapse is Connected.
   NOTE: When your iPhone Camera is connected to KISS Controller you will not be able to use the keyboard on your iPhone Camera.

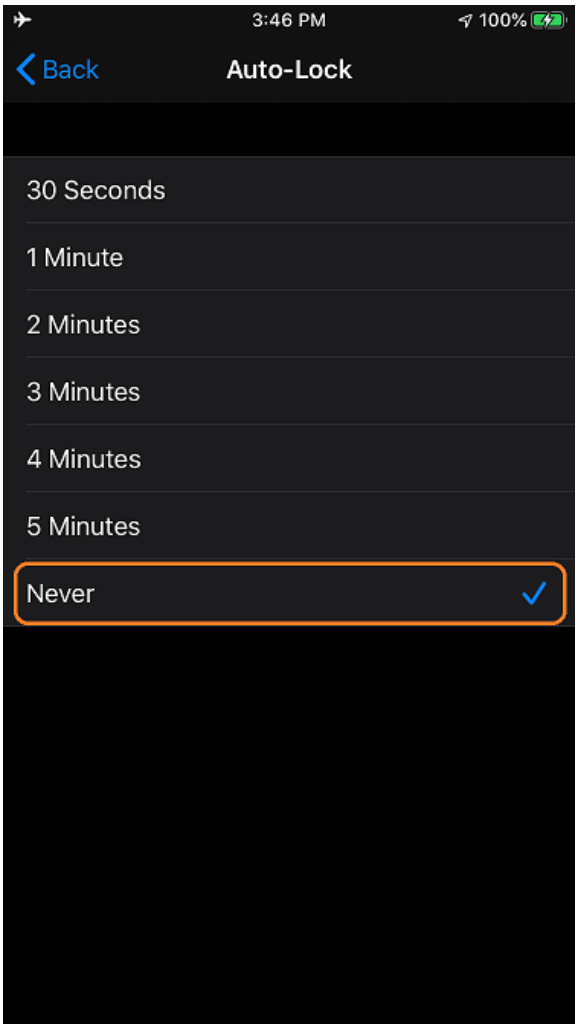


Camera Always ON

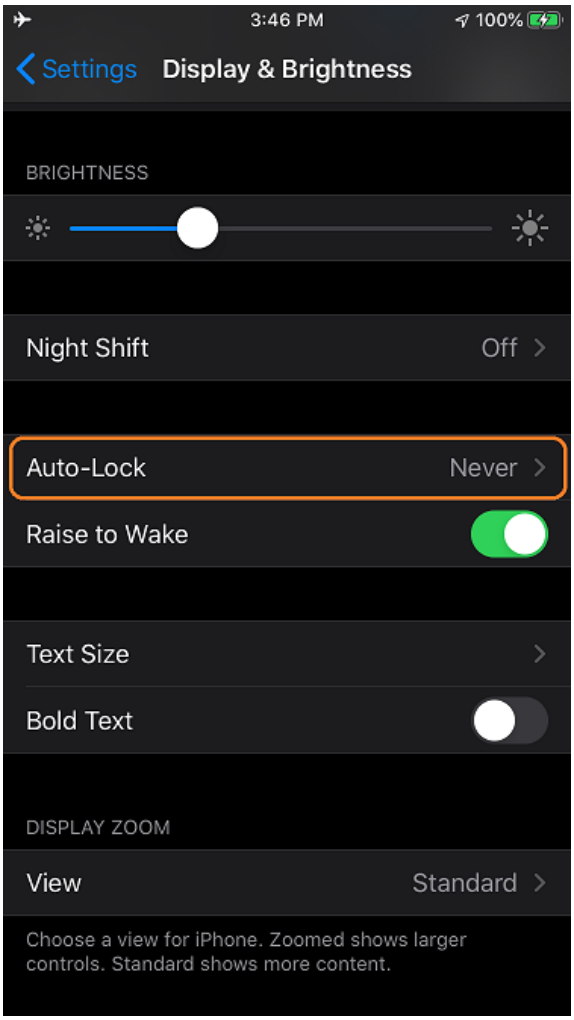
1. Go to Settings -> Display & Brightness
   Click on “Auto-Lock”



2. Select “Never” and click “< Back”
   NOTE: If your iPhone is setup with Stanford Device Management you will not be able to set “Never”.

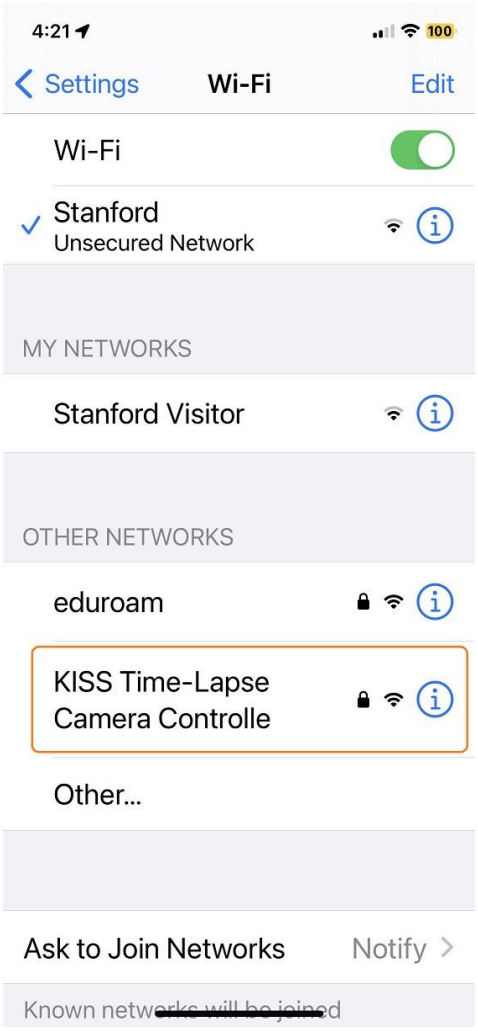


3. Verify that “Never” is shown for “Auto-Lock”

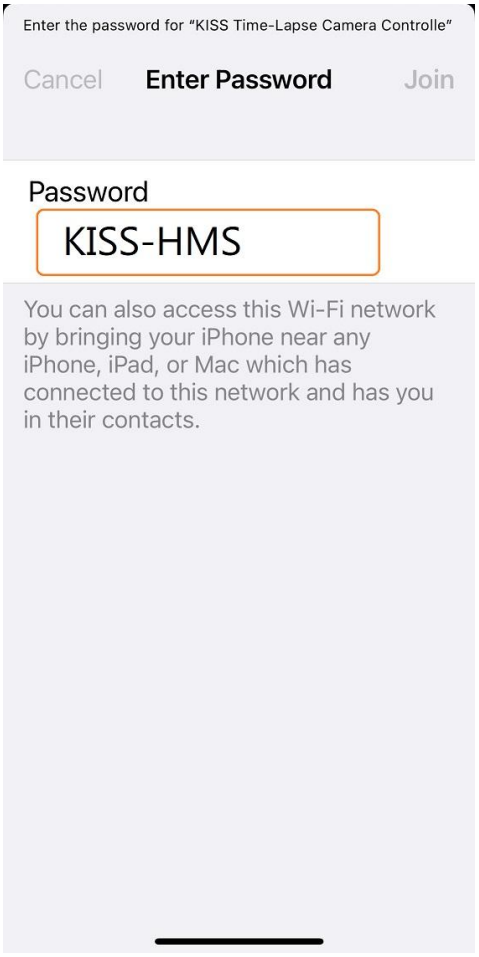


KISS WiFi Access Point

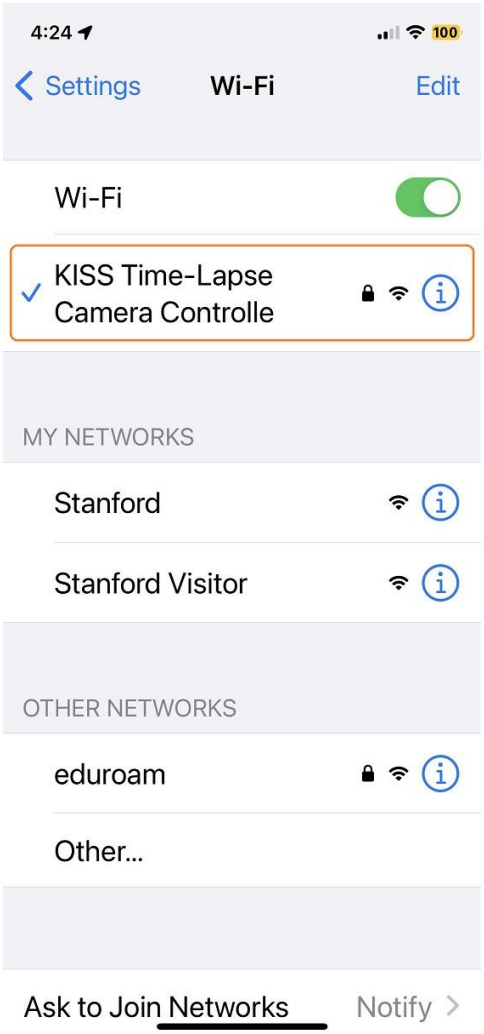
1. Go to Settings -> Wi-Fi
   Under OTHER NETWORKS you should see
   KISS Time-Lapse Camera Controller



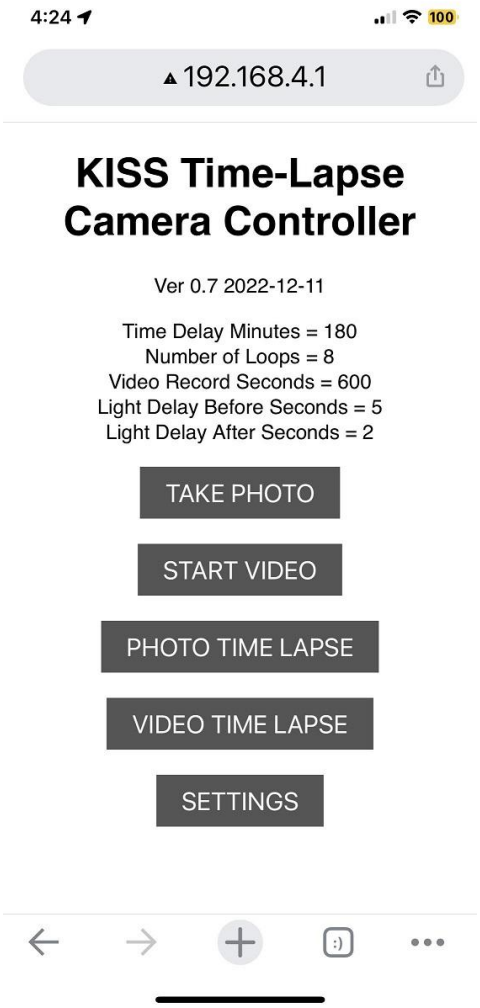
2. Select the “KISS Time-Lapse Camera Controller” and you will be prompted for a password. Enter “KISS-HMS”



3. You should see a connect to the “KISS Time-Lapse Camera Controller” NOTE: When you are connected to the KISS Controller you will not have access to the Internet



4. Open a browser and navigate to the URL:
   192.168.4.1
   This will open up the KISS User Interface

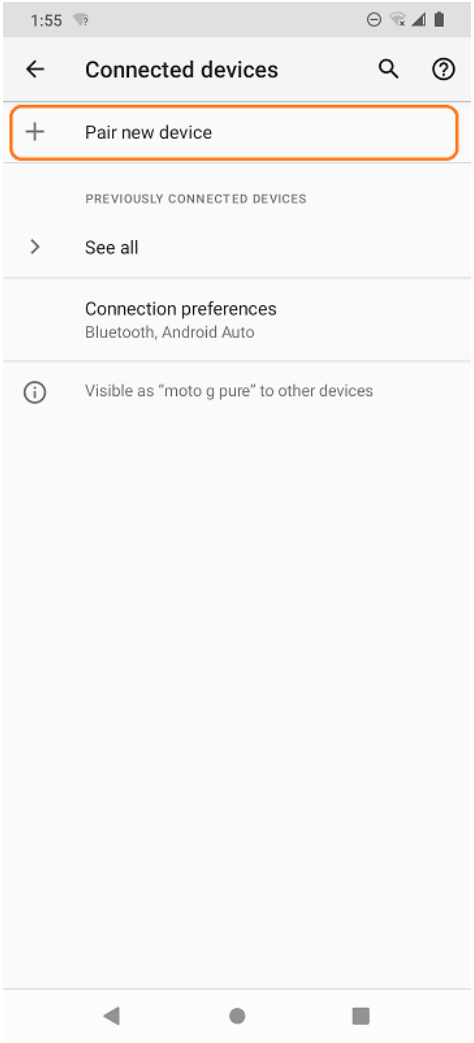


Android

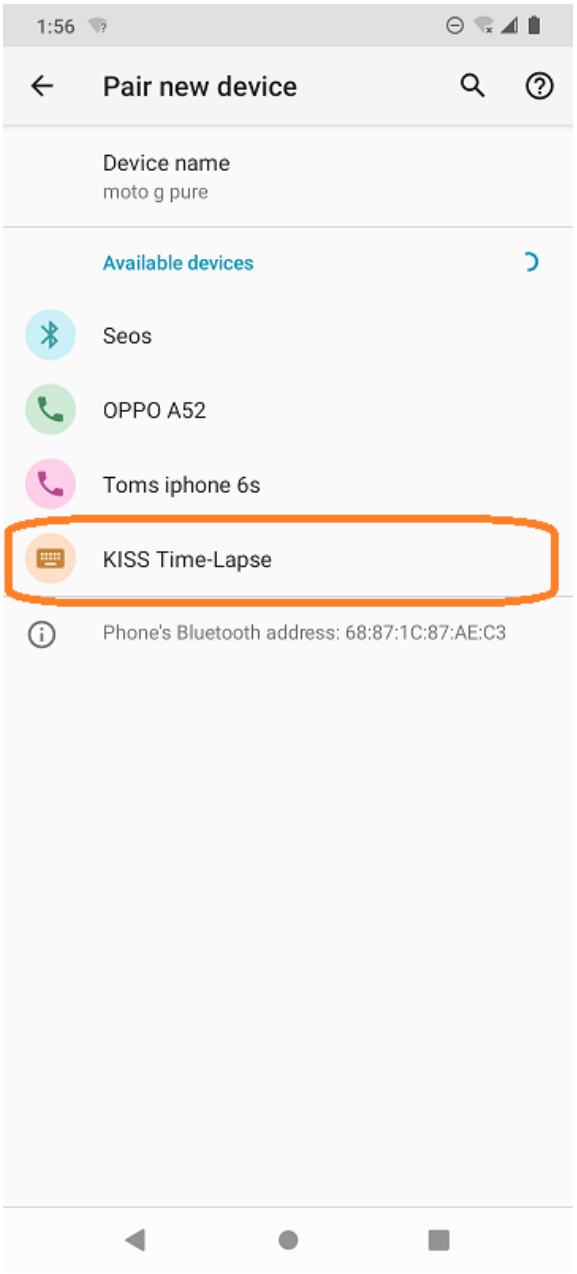
Camera Setup

Bluetooth for Camera Shutter

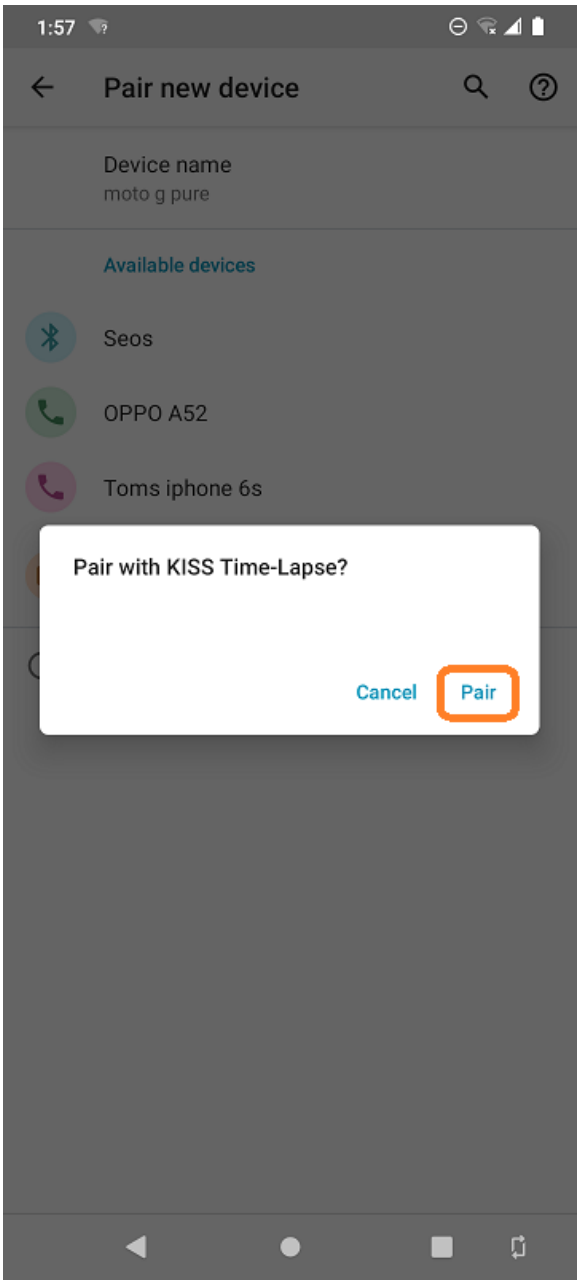
1. Go to Settings -> Connected Devices
   Select the option to “Pair new device”.



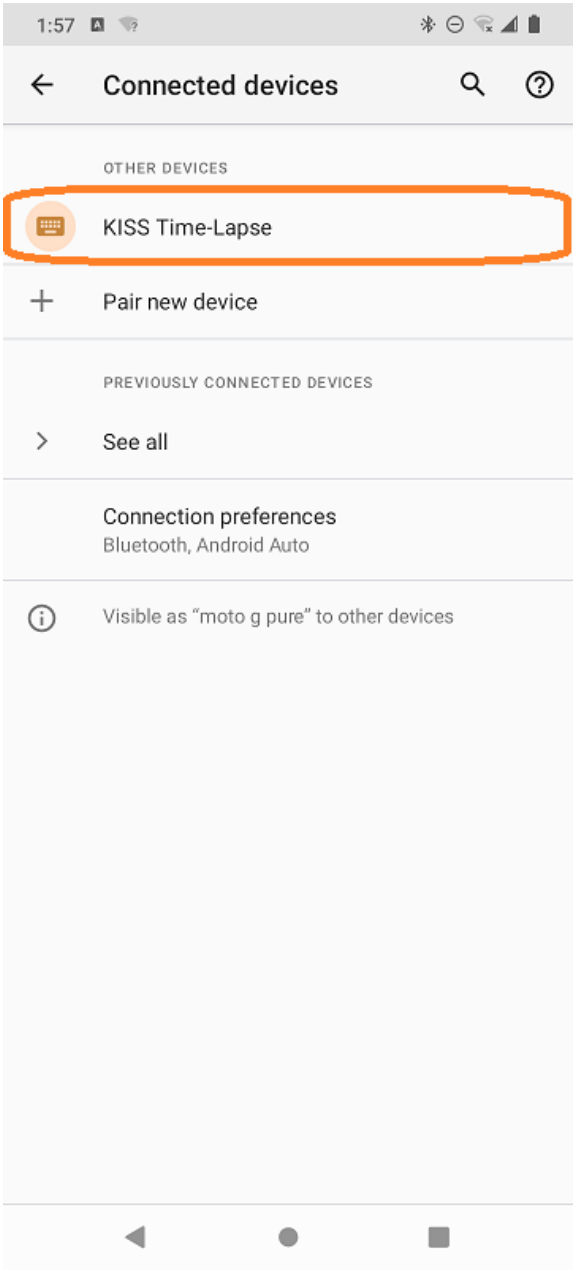
2. Select the KISS Time-Lapse and you will be prompted with a pairing request from the KISS Microcontroller.



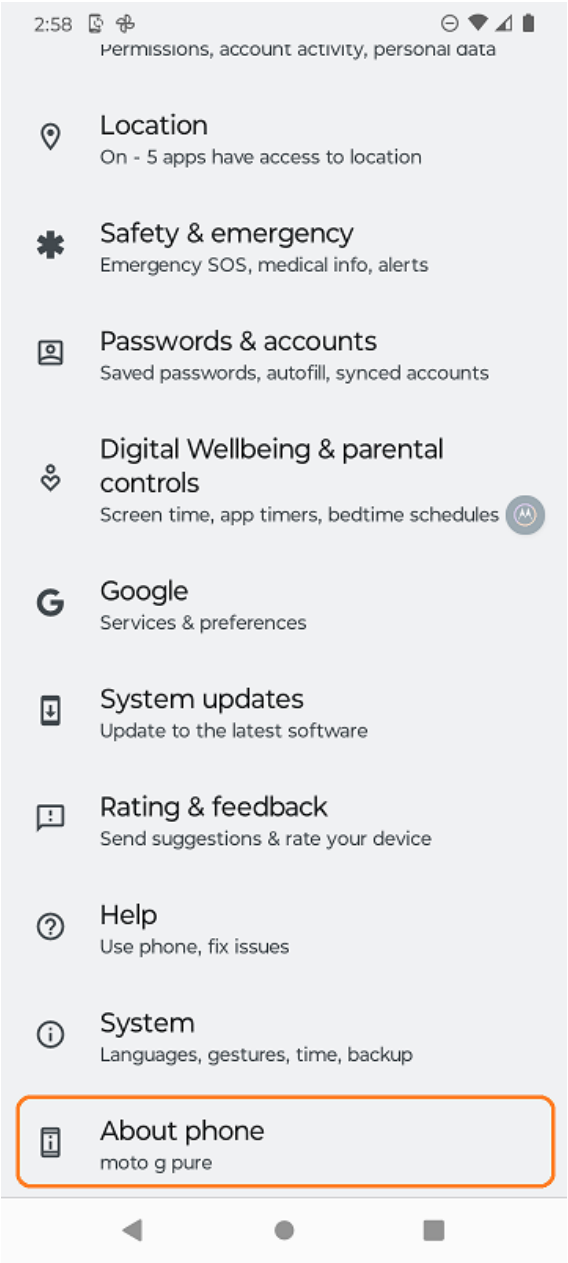
3. Click on the “Pair” button.



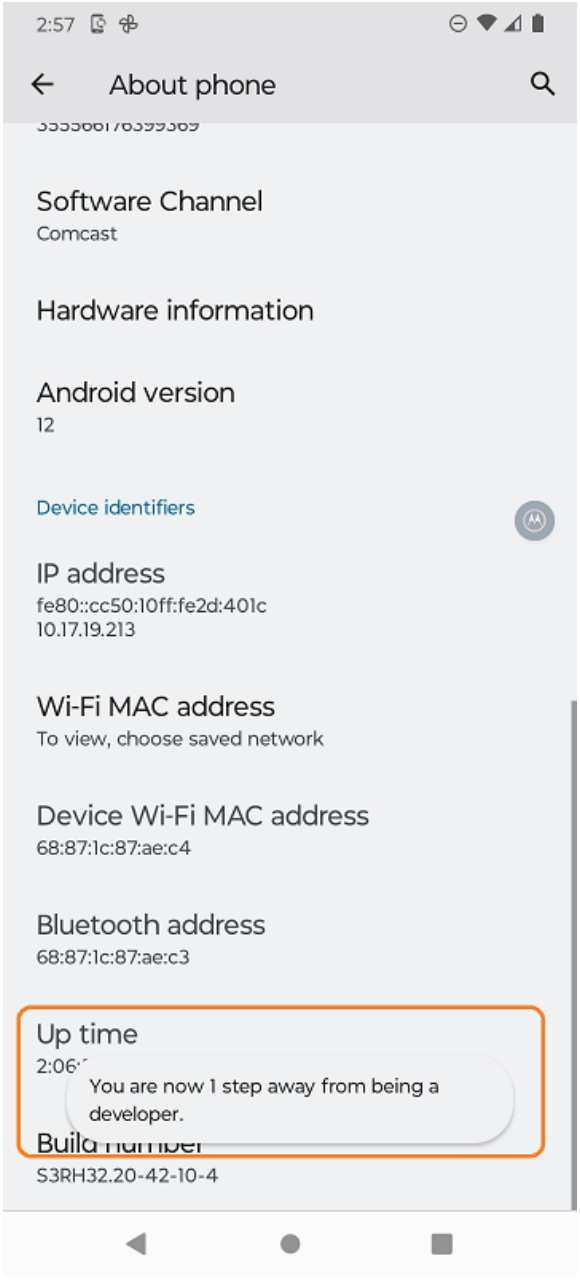
4. When the pairing is completed you will see that the KISS Time-Lapse is Connected.
   NOTE: When your Android Camera is connected to KISS Controller you will not be able to use the keyboard on your Android Camera.



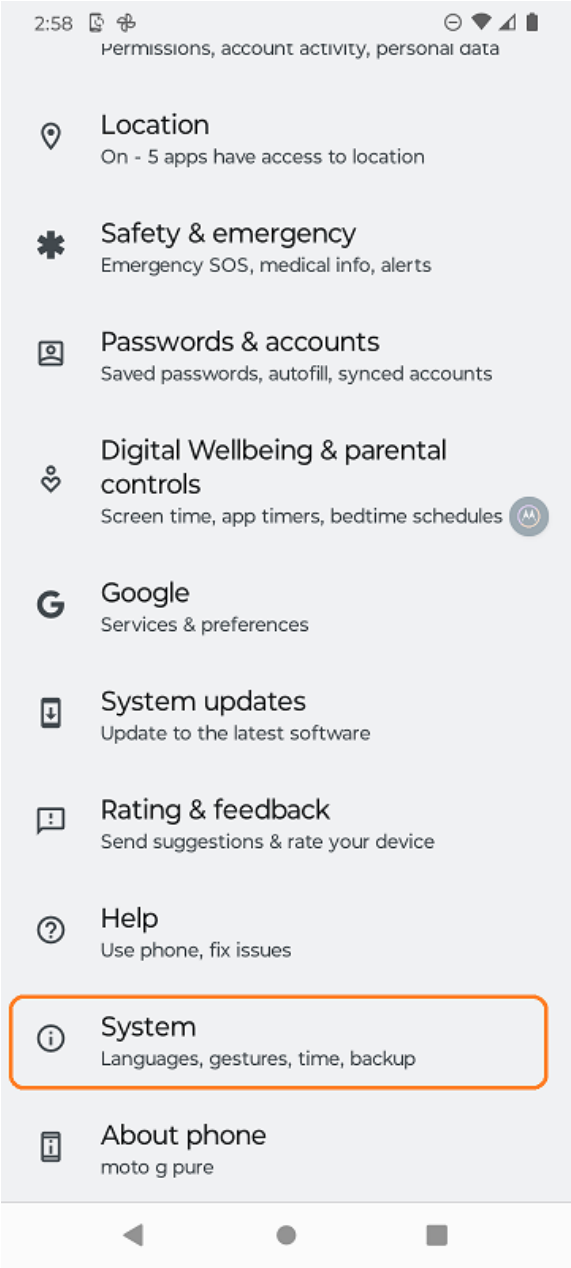
1. Go to Settings -> About Phone



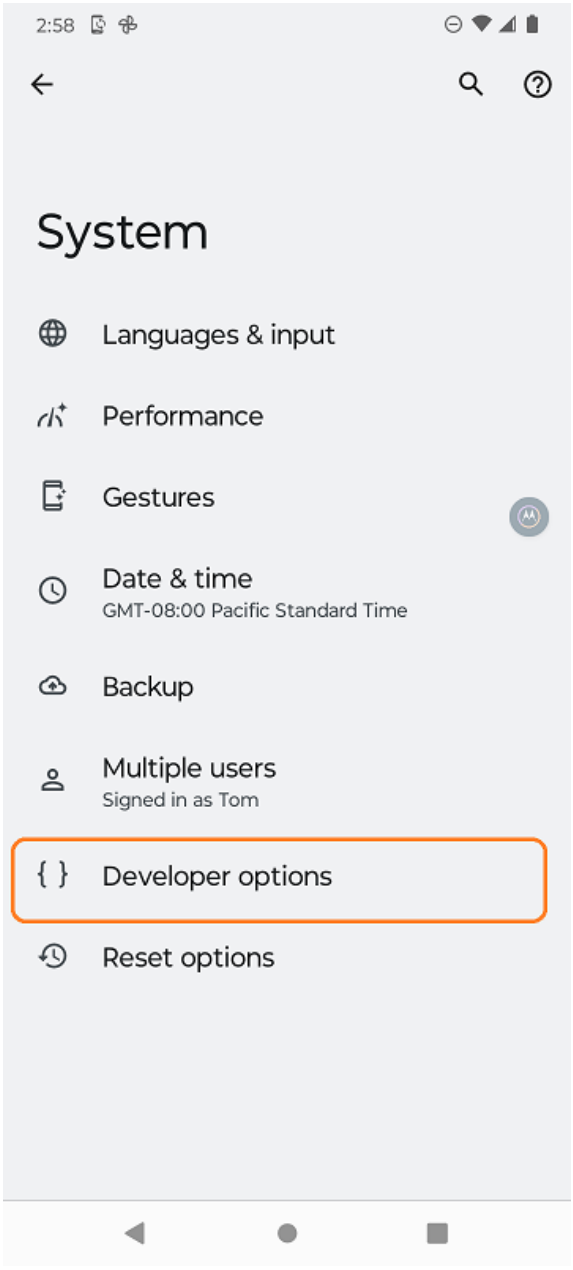
2. Tap on the Build number repeatedly until the "You Are now a Developer" message is displayed.



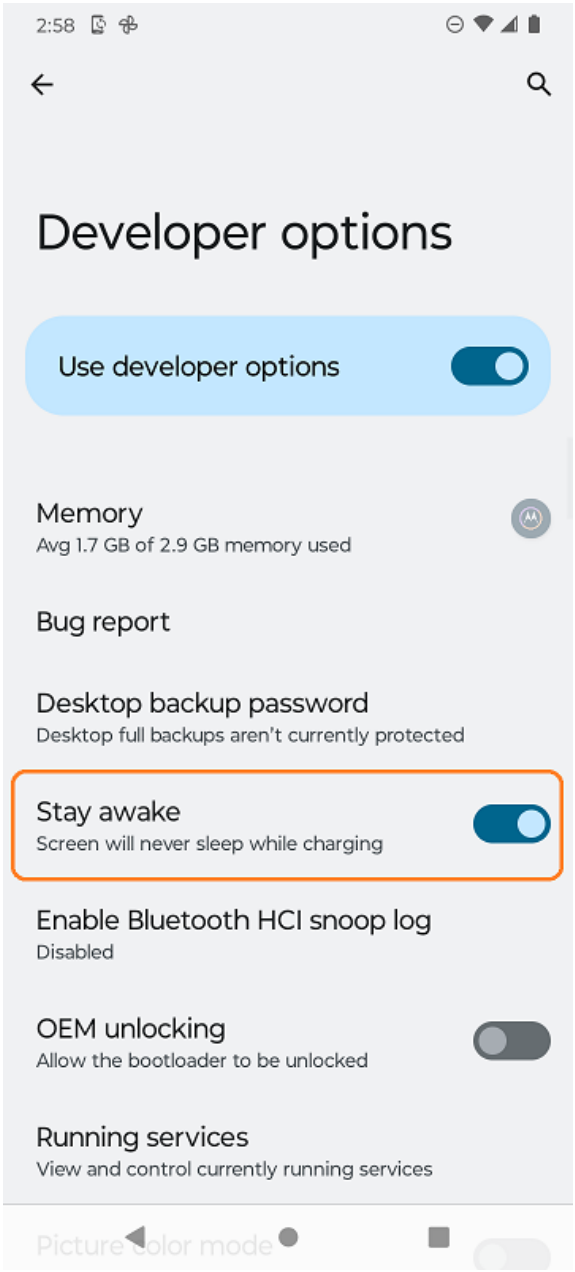
3. Go to Settings -> Controller



4. Go to Controller -> Developer Options



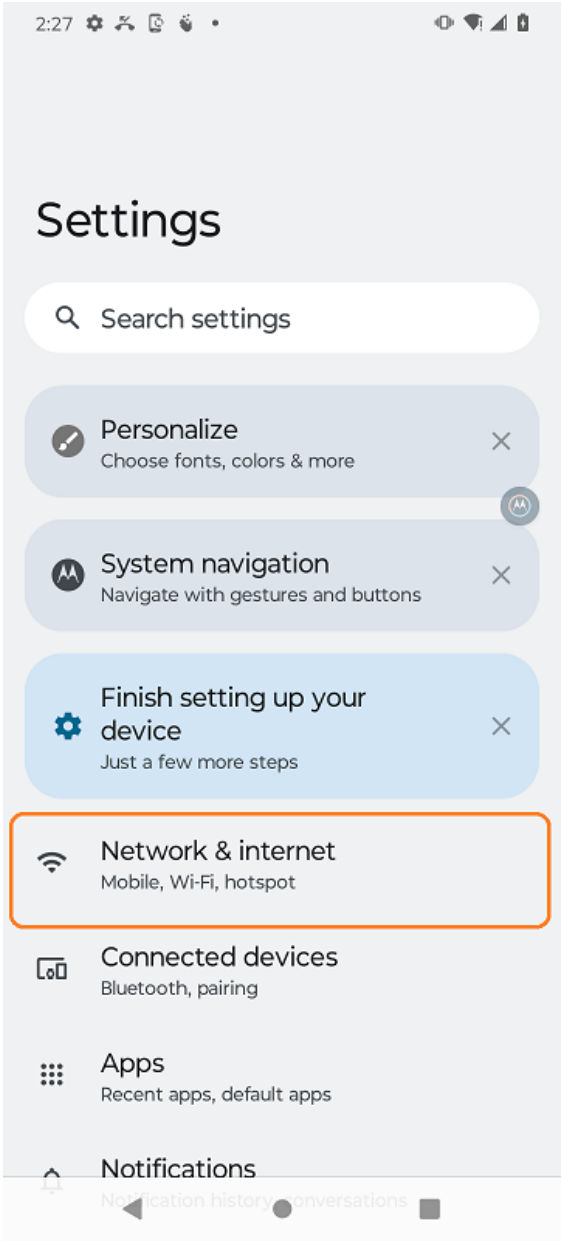
5. Toggle Stay Awake to enabled



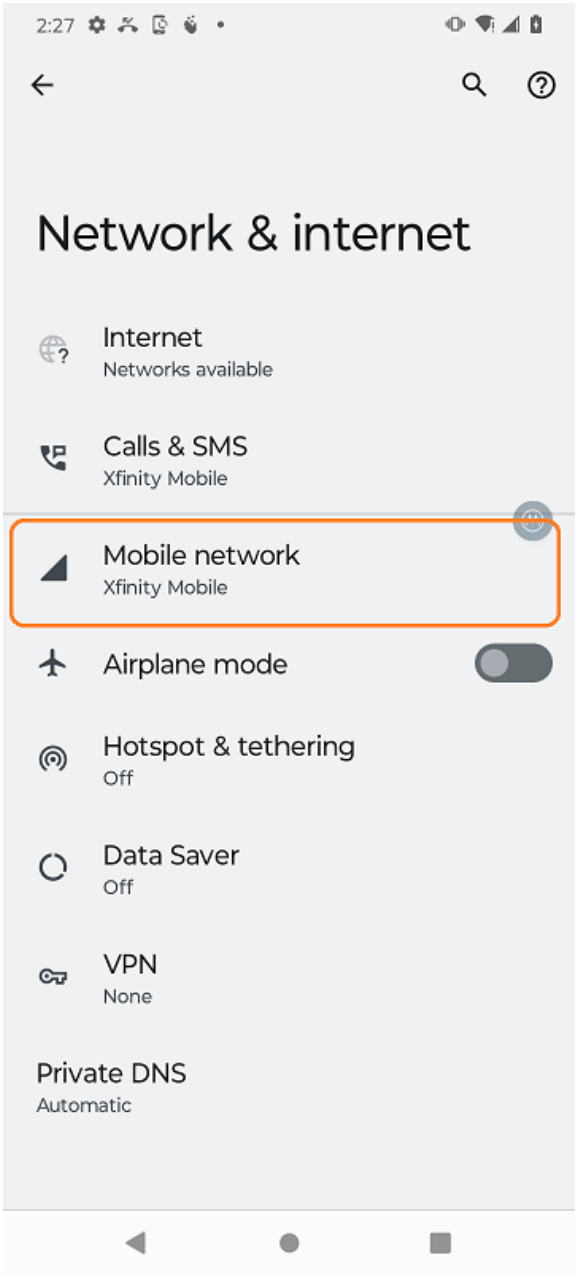
Android Allow Data Roaming

This set is required for Android phones because the WiFi Access Point does not have an Internet Connection.

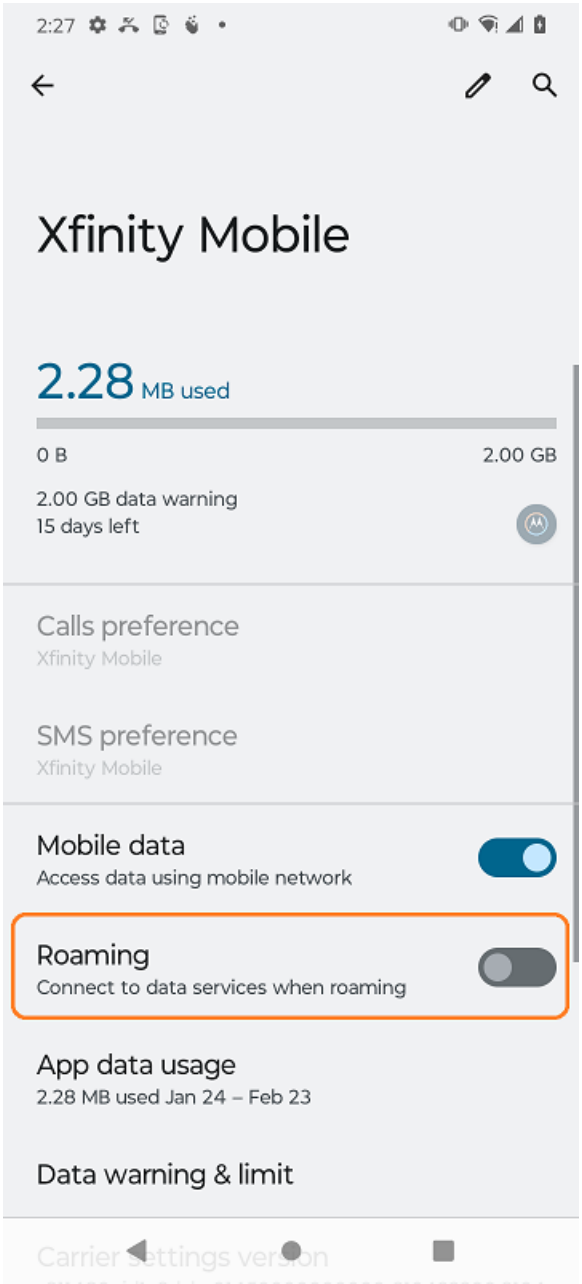
1. Go to Settings -> Network & Internet



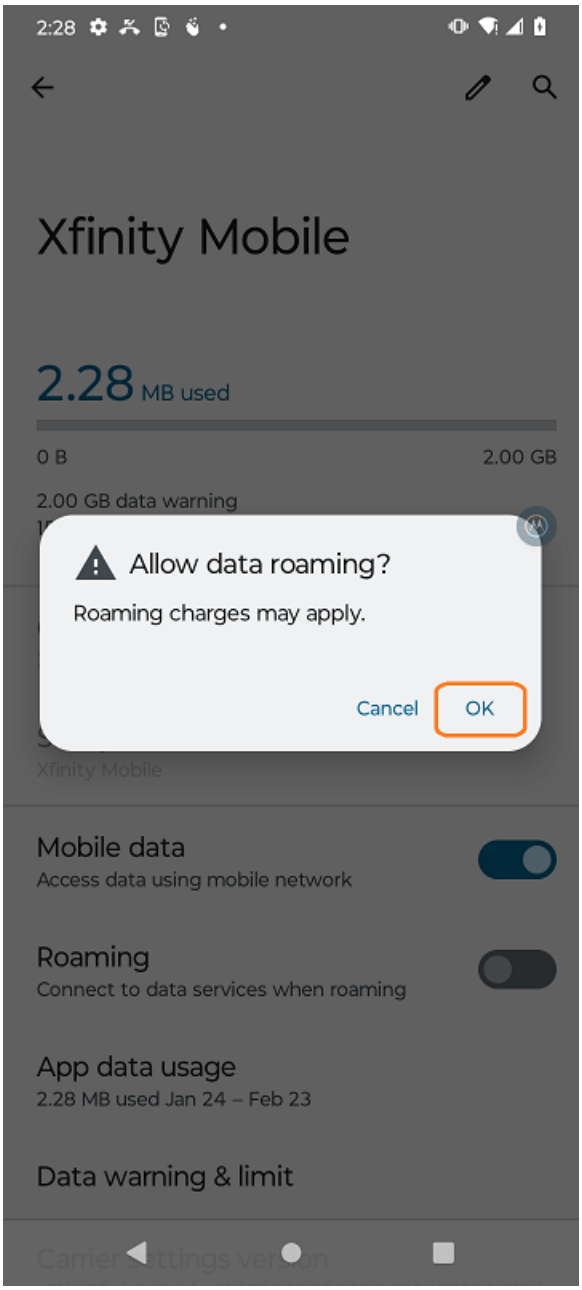
2. Go to Mobile Network



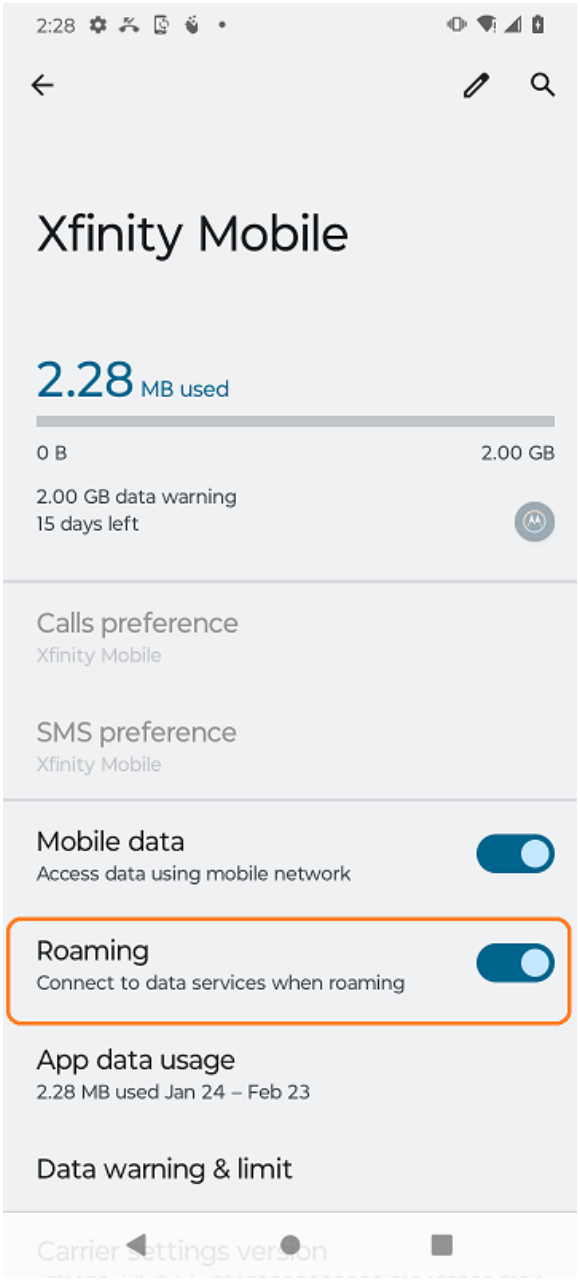
3. Go to Roaming



4. Click OK


2:28
Xfinity Mobile
2.28 MB used
0 B
2.00 GB
2.00 GB data warning
Allow data roaming?
Roaming charges may apply.
Cancel
OK
Xfinity Mobile
Mobile data
Access data using mobile network
Roaming
Connect to data services when roaming
App data usage
2.28 MB used Jan 24 – Feb 23
Data warning & limit

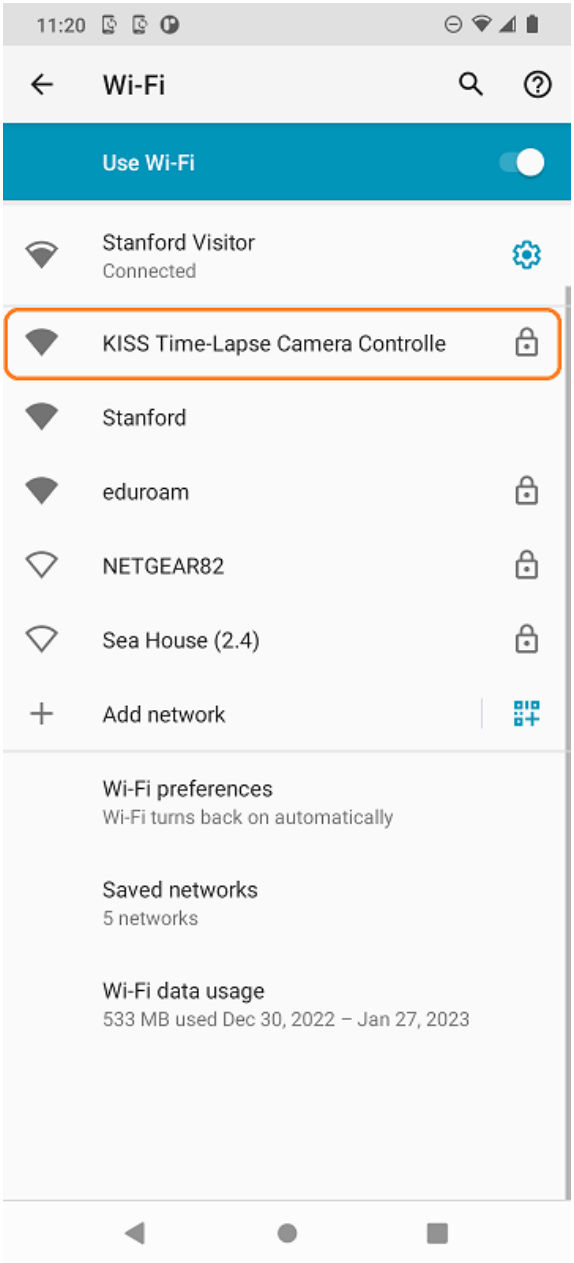
5. You should see Roaming enabled



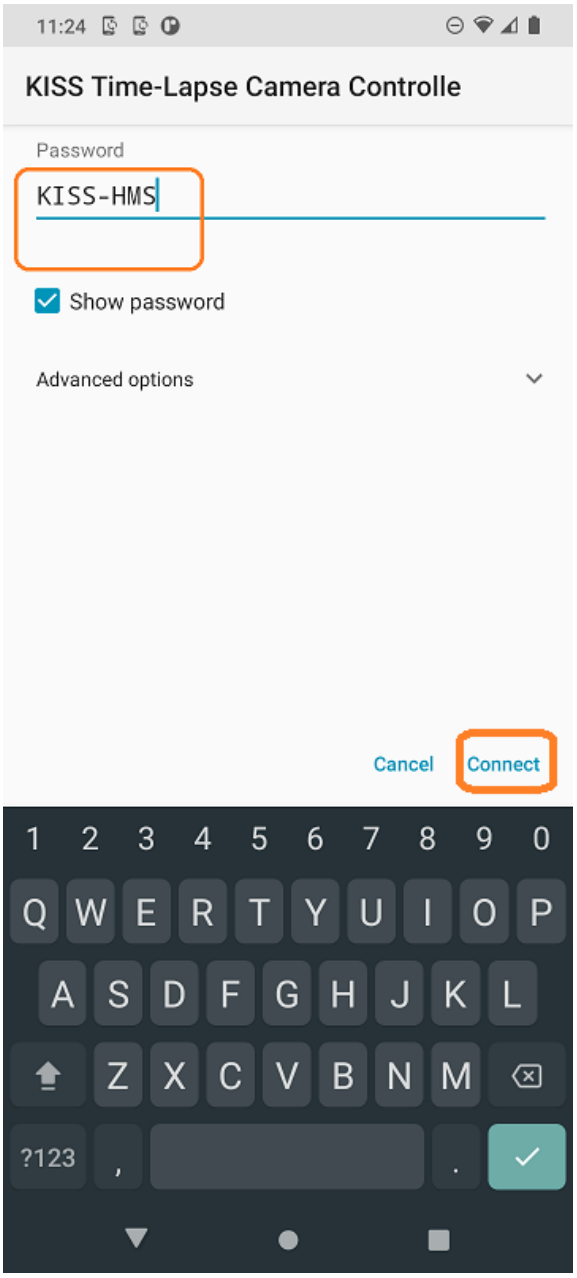
Android User Interface

KISS WiFi Access Point

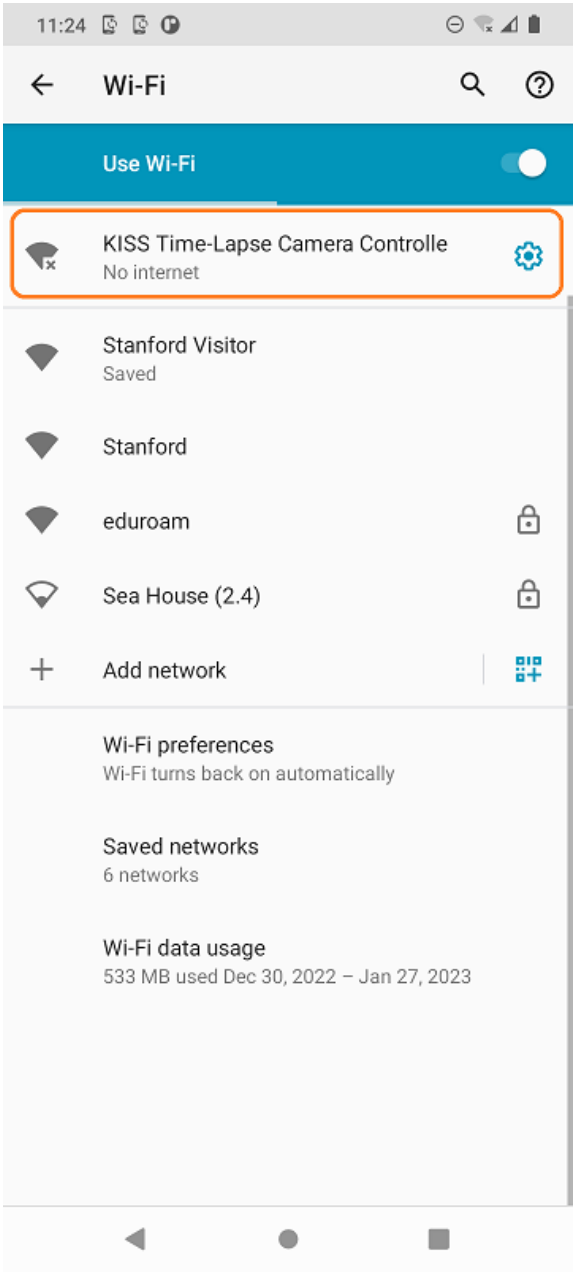
1. Go to Settings -> Network & Internet
   You will see the KISS Time-Lapse Camera Controller



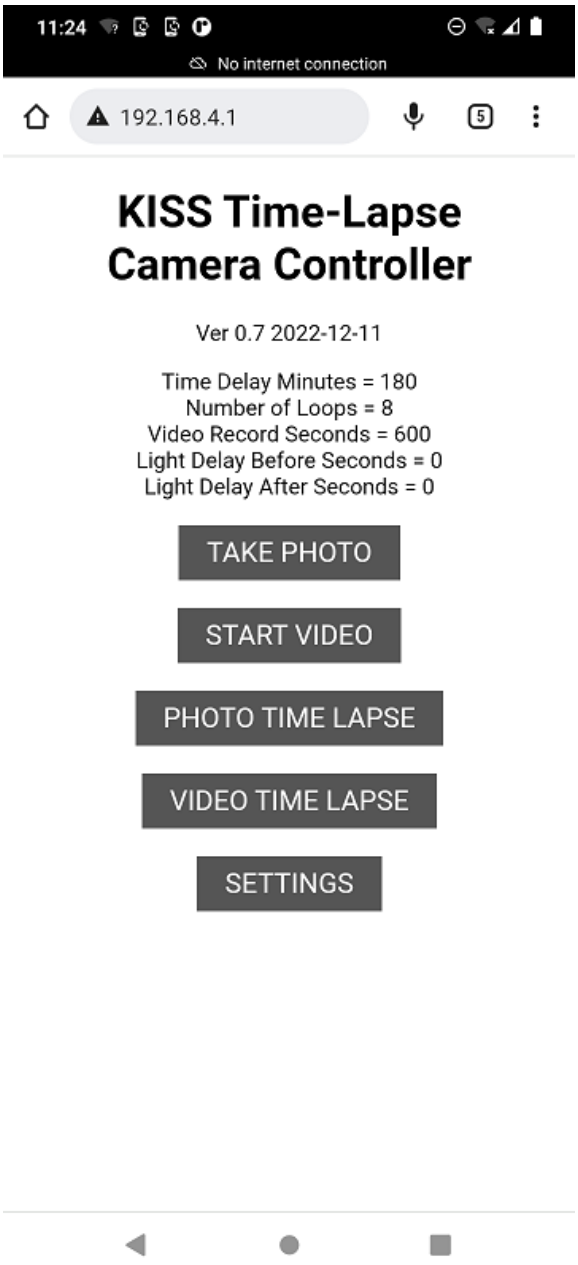
2. Select the “KISS Time-Lapse Camera Controller” and you will be prompted for a password. Enter “KISS-HMS” and then “Connect”



3. You should see a connect to the “KISS Time-Lapse Camera Controller”
   NOTE: When you are connected to the KISS Controller you will not have access to the Internet



4. Open a browser and navigate to the URL:
   192.168.4.1
   This will open up the KISS User Interface



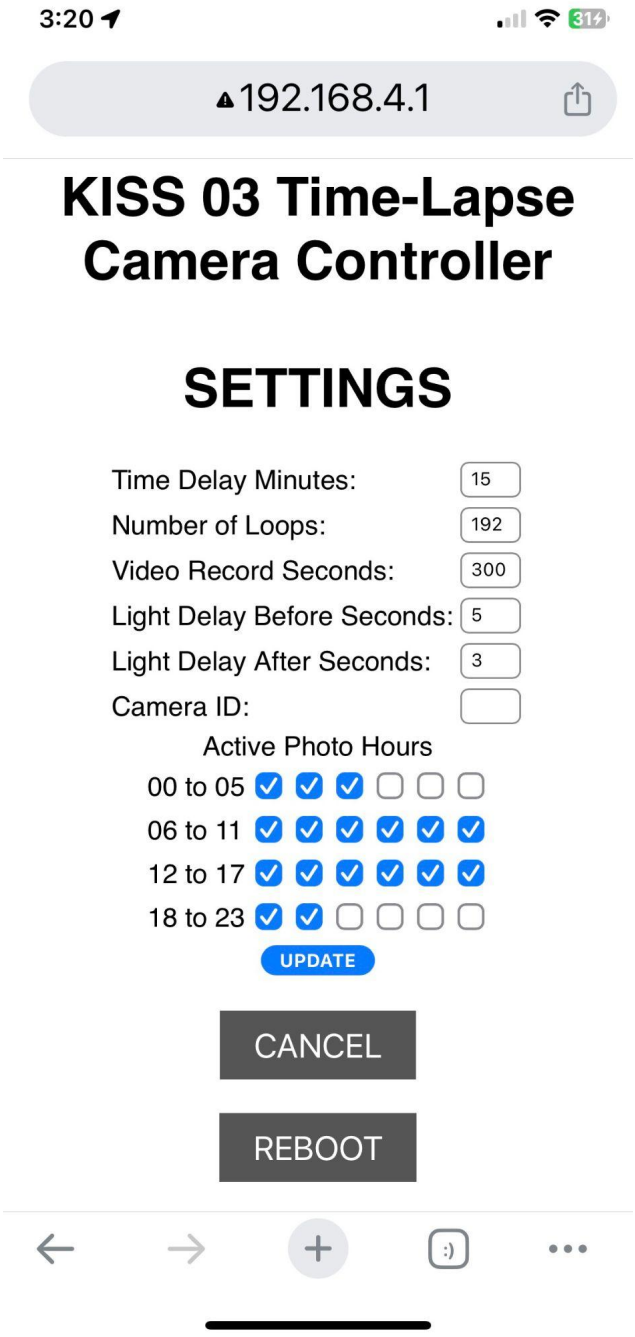
KISS Controller Operation

When both the SmartPhone Camera and SmartPhone User Interface setup have been completed you should see the following on your user interface. Note that if your laptop supports a bluetooth connection, you can also control the KISS Controller from your laptop.

Settings

There are 5 Settings which are used to control the KISS Controller Operation:

Settings Display



Time Delay Minutes

This is the delay between photos that are taken to produce a time-lapse video.

Number of Loops

This is the number of times to repeat the loop of taking a photo and then delaying.

Video Record Seconds

This value is used when you choose to do a Video Time Lapse and specifies the number of seconds for recording a video during each loop. Note that you must manually place your camera in the Video mode.

Light Delay Before Seconds

This is the delay after the light is turned on before a photo is taken. It is usually required to allow the SmartPhone camera to complete its auto focusing.

Light Delay After Seconds

This is the delay after a photo is taken before the light is turned off. It is only required if there is a delay between sending a command to take the photo and having the photo operation take place.

CameraID

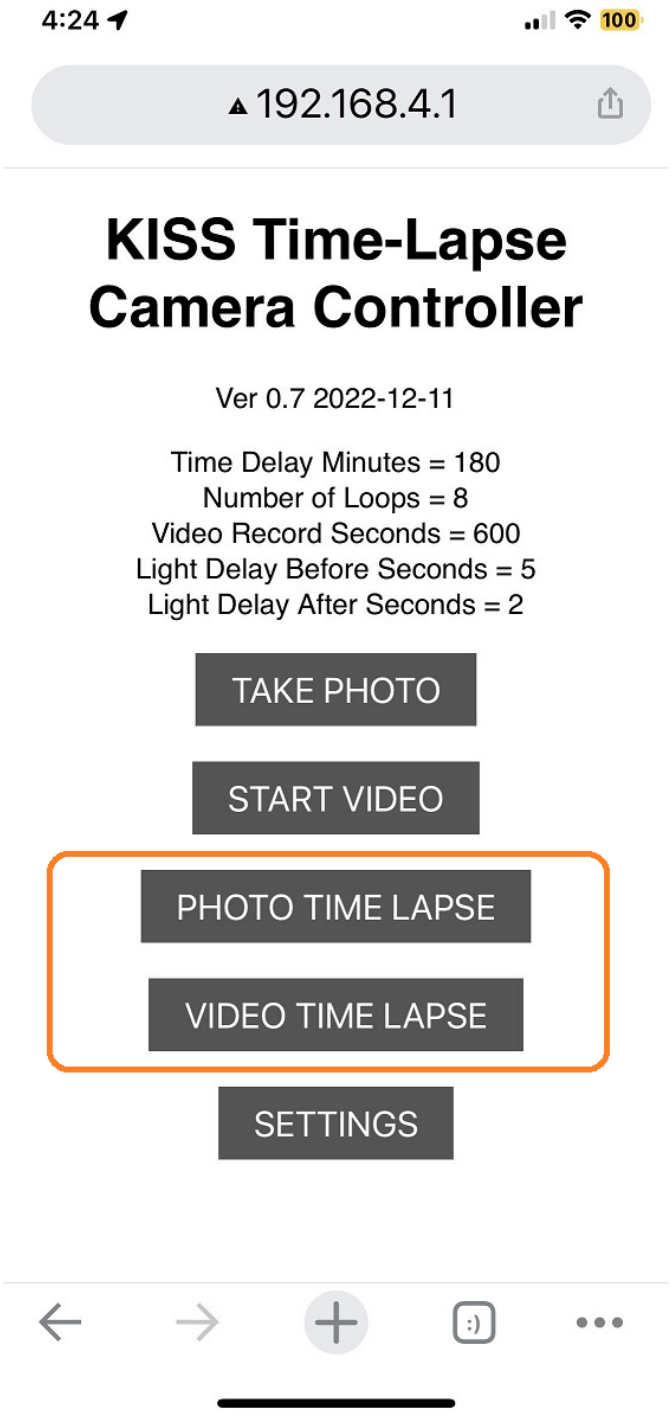
If you have more than one KISS Controller you will need to have a different CamerID for each Smartphone. Note that this setting affects both the WiFi Access Point name and the Bluetooth keyboard name. When you change this setting you should power off the KISS Controller and “forget” the former WiFi Access Point and Bluetooth keyboard.

Active Photo Hours

Turn on the check for each of the hours that you want to have recordings.

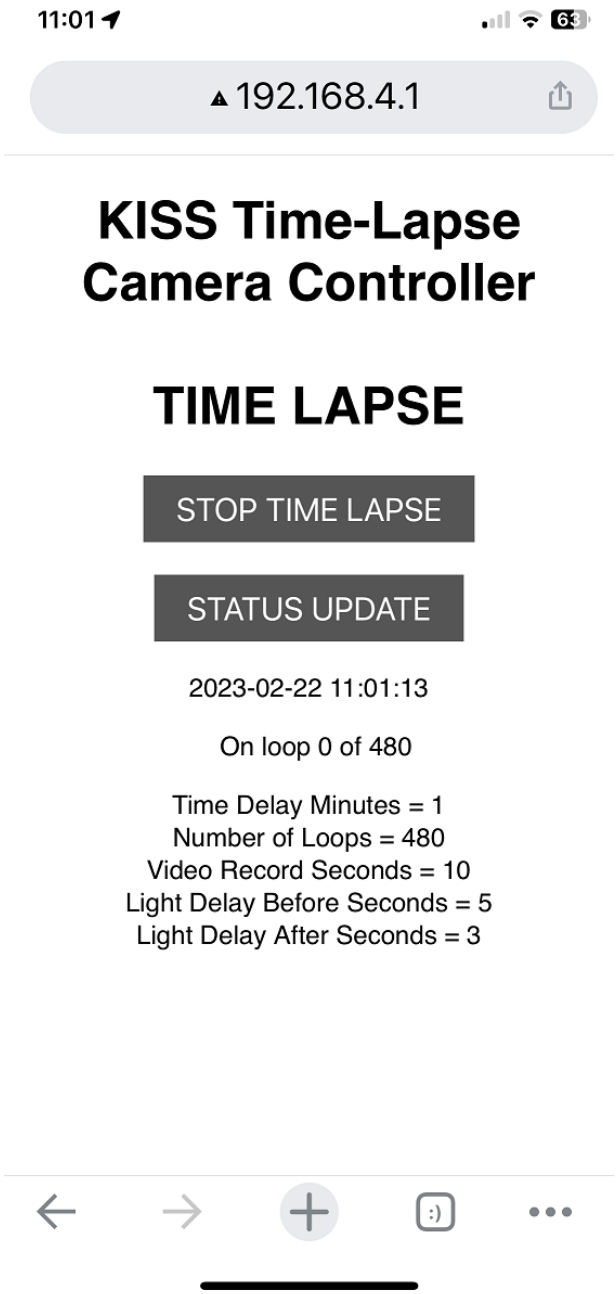
Photo Time Lapse and Video Time Lapse

After selecting the Settings for your Time Lapse using the Settings operation, begin the Time Lapse by clicking on either the Photo Time Lapse or the Video Time Lapse buttons:



After clicking on a button you will see a Time Lapse status screen. This screen shows the loop number and the total number of loops as well as the settings. Clicking on “Status Update” will update the screen including the current time. You can abort the Time Lapse by clicking the “Stop Time Lapse”.

Note that it is recommended to click the “Status Update” before disconnecting your Smartphone from the KISS system WiFi Access Point. Then when you reconnect to the KISS system you can directly perform a Status Update.



KISS Controller Construction

Construction of the KISS Controller includes printing 3D parts, assembling the components, and wiring the Microcontroller.

3D Printed Parts

The 3D printed parts are all Included as STL files and can be printed with single extruder filament printers with no requirements for support structure or brims.

Microcontroller Enclosure and Bracket

There are 5 printed parts of the Microcontroller enclosure and bracket:

- Microcontroller enclosure with attachment bracket
- Cover for the Microcontroller including RST and BOOT buttons
- Bolt
- Washer
- Wing Nut

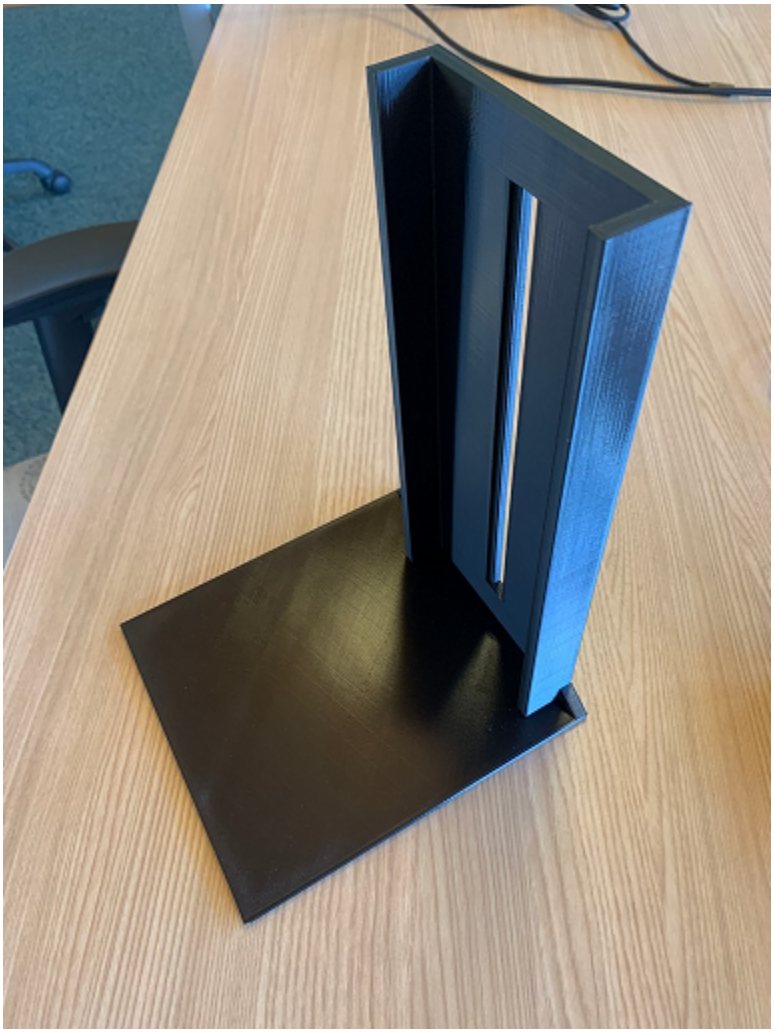
Smartphone Holder and Bracket

There are 4 components of the Smartphone holder and bracket:

- Smartphone holder with attachment bracket
  Rubber bands are used do affix the camera to the holder
- Bolt
- Washer
- Wing Nut

Stand for Microcontroller and Smartphone Brackets

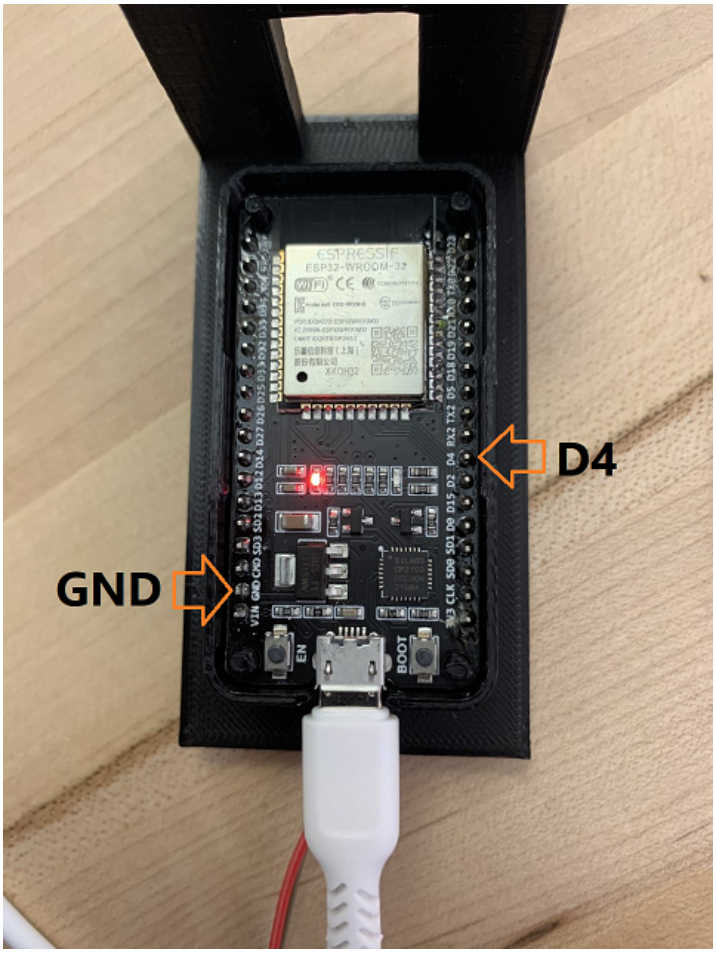
The stand is used to attach the vertically adjustable Smartphone and Microcontroller brackets. The full scale stand requires the ability to print an object of dimensions: 185 x 185 x 250 mm. You may need to scale the stand.



Parts List

ESP32-WROOM-32 Microcontroller

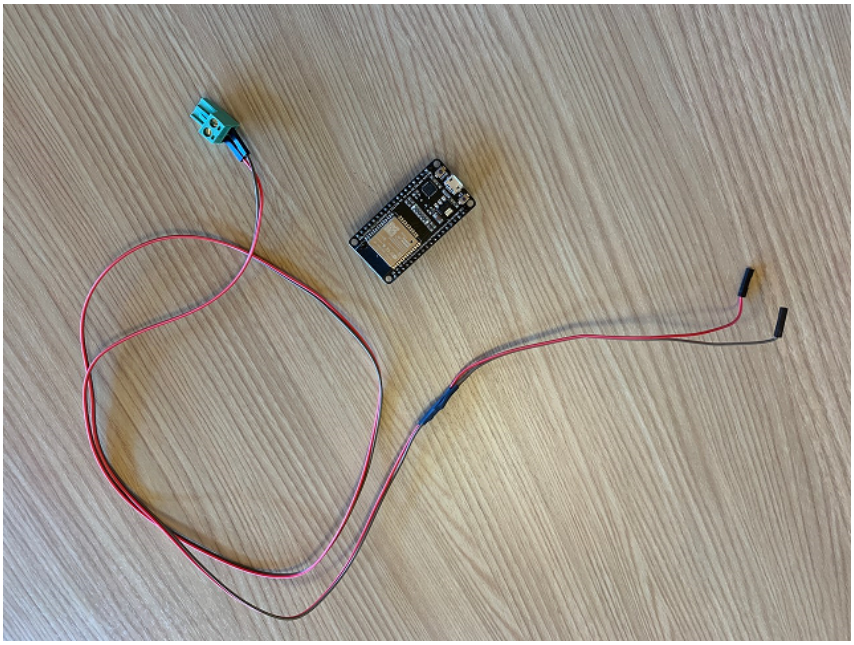
This Microcontroller provides a powerful, generic Wi-Fi + Bluetooth® + Bluetooth LE MCU module. It is used by the KISS Controller to provide a Bluetooth interface for the Smartphone camera to operate the shutter and also as the User Interface running on the WiFi Access Point.



Controllable Four Output Power Relay Module

Wires

Male to Female jumper wires of suitable length to connect the Four Output Relay Module to the Microcontroller. The male end connects into the input for the relay module. The female end connects to the pins of the ESP32 WROOM Microcontroller.

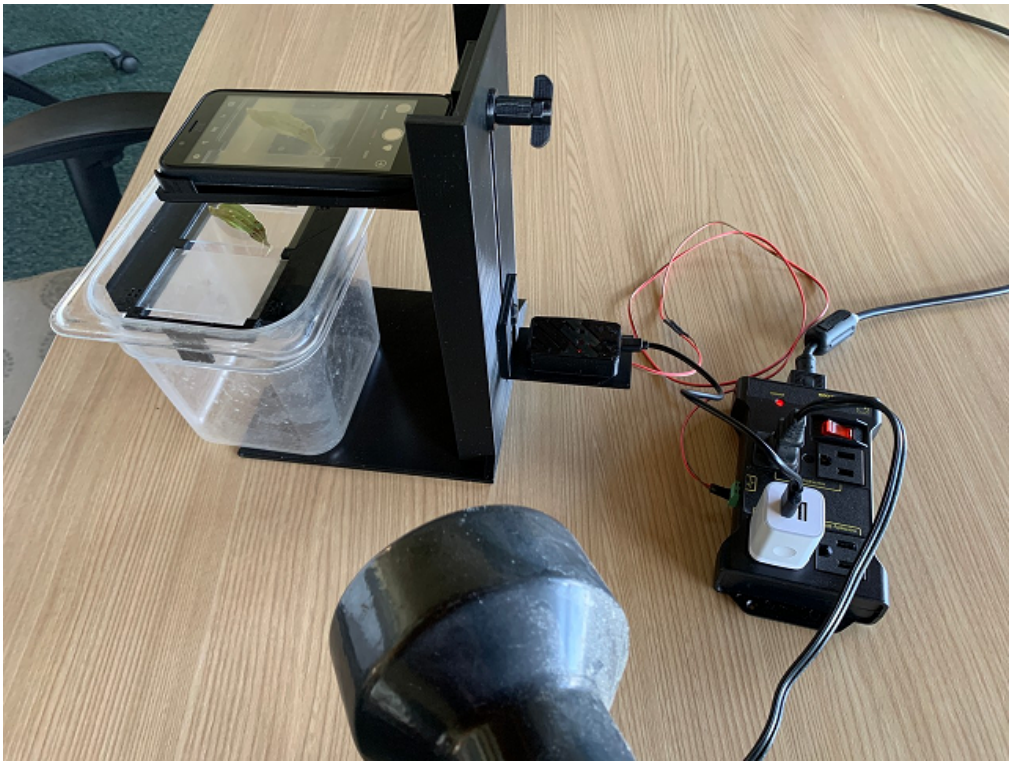


KISS Controller Assembly

Position the Smartphone holder on the stand and secure it with the wing nut, washer and bolt. Locate it at a proper height above the subject that you are photographing so that it is in focus. For lengthy time-lapse sequences you will need to connect the Smartphone to the USB power adapter. If you do not have a USB power adapter with two USB slots you will need two USB power adapters. In this case, plug the Smartphone into the USB power adapter that is plugged into the socket labeled “normally ON”.

Attach the Microcontroller bracket to the back side of the stand and secure it with the wing nut, washer and bolt. The Microcontroller has two wires which are then connected to control the Four Output Relay Module. A micro USB cable is used to connect the Microcontroller to a USB power adapter. If you do not have a USB power adapter with two USB slots you will need two USB power adapters. In this case, make sure that you plug the Microcontroller into the socket labeled “always ON”.

The last step in the KISS Controller Assembly is to plug your light source into the socket labeled “normally OFF”. Note that there are two sockets with this label allowing you to connect two light sources at different positions to illuminate your subject.



Arduino (ESP32 WROOM) Source Code for KISS

```
/*************************************************************
************
  Dual-Axis Smartphone Slider
  KISS Time-Lapse Camera Controller

  Original Code:  2022-11-01

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

  To Do:

  Notes:
    I switched to noInterrupts() and interrupts() around
critical sections


*************************************************************
***********/

#define PROGRAM "KISS Time-Lapse Camera Controller"
#define VERSION "Ver 0.9 2023-03-09"

#define DEBUG_OUTPUT  1

// NOTE: this define is required inside BleKeyboard.h to work
on iPhones and Androids!
//#define USE_NIMBLE

#include <BleKeyboard.h>
//https://github.com/T-vK/ESP32-BLE-Keyboard
//BleKeyboard bleKeyboard("KISS 00 Time-Lapse BLE Kybd",
"Hopkins Miller Fab Lab", 100);
BleKeyboard bleKeyboard("", "Hopkins Miller Fab Lab", 100);

#include <EEPROM.h>
// EEPROM.write(address, value);
// EEPROM.commit();
```

```
// EEPROM.read(address);

// EEPROM state variables
#define EEPROM_SIZE 20
#define EEPROM_SIGNATURE                    0 // 000-003
Signature 'KISS'
#define EEPROM_NUMBER_OF_LOOPS              4 // 004-005
iNumberOfLoops
#define EEPROM_TIME_DELAY_MINUTES           6 // 006-007
iTimeDelayMinutes
#define EEPROM_VIDEO_RECORD_SECONDS         8 // 008-009
iVideoRecordSeconds
#define EEPROM_LIGHT_DELAY_BEFORE_SECONDS  10 // 010-011
iLightDelayBeforeSeconds
#define EEPROM_LIGHT_DELAY_AFTER_SECONDS   12 // 012-013
iLightDelayAfterSeconds
#define EEPROM_CAMERA_ID                   14 // 014-015
cCameraID

#define LIGHT_CONTROL_PIN 4

#define NUMBER_OF_LOOPS             2
#define TIME_DELAY_MINUTES          1
#define VIDEO_RECORD_SECONDS        60
#define LIGHT_DELAY_BEFORE_SECONDS  5
#define LIGHT_DELAY_AFTER_SECONDS   2

#define MAX_OPERATIONS          10
#define OP_PHOTO                1
#define OP_STARTVIDEO           2
#define OP_STOPVIDEO            3
#define OP_STARTPHOTOTIMELAPSE  4
#define OP_STARTVIDEOTIMELAPSE  5
#define OP_STOPTIMELAPSE        6
#define OP_SETTINGS             7
#define OP_CANCEL               8
#define OP_REBOOT               9
#define OP_STATUSUPDATE        10

#define TIMEDELAYMINUTES        "TIMEDELAYMINUTES"
#define NUMBEROFLOOPS           "NUMBEROFLOOPS"
#define VIDEORECORDSECONDS      "VIDEORECORDSECONDS"
#define LIGHTDELAYBEFORESECONDS "LIGHTDELAYBEFORESECONDS"
#define LIGHTDELAYAFTERSECONDS  "LIGHTDELAYAFTERSECONDS"
#define CAMERAID                "CAMERAID"
```

```
// NOTE: I switched to noInterrupts() and interrupts() around
critical sections

static int iOperations[MAX_OPERATIONS];
static int iOpIn = 0;
static int iOpOut = 0;
static int iOpCount = 0;

#include <WiFi.h>

#include <ESPAsyncWebServer.h>

const char* wifi_network_ssid     = "Stanford";
const char* wifi_network_password =  "";

char soft_ap_ssid[]               = "KISS 00 Time-Lapse
Camera";
//                                   0123456
const char *soft_ap_password      = "KISS-HMS";

// Set web server port number to 80
AsyncWebServer server(80);
//WiFiServer server(80);

const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = -(8*3600);
int   daylightOffset_sec = 3600;  //  = 0;

#define STATUSUPDATE_INITIAL  0
#define STATUSUPDATE_RUNNING  1
#define STATUSUPDATE_FINISHED 2

static int iStatusUpdateState = STATUSUPDATE_INITIAL;

#define NO_TIMELAPSE    0
#define PHOTO_TIMELAPSE 1
#define VIDEO_TIMELAPSE 2

static int iTimeLapse = NO_TIMELAPSE;
static bool bTimeLapseFinished = false;

static int iTimeDelayMinutes = TIME_DELAY_MINUTES;
static int iNumberOfLoops = NUMBER_OF_LOOPS;
static int iLoopCounter = 1;
```

```
static int iVideoRecordSeconds = VIDEO_RECORD_SECONDS;
static long lVideoRecordMilliseconds = 0;
static int iLightDelayBeforeSeconds =
LIGHT_DELAY_BEFORE_SECONDS;
static int iLightDelayAfterSeconds = LIGHT_DELAY_AFTER_SECONDS;
static char cCameraID[3] = {'0','1','\0'};   // 01

char sKeyboardName[] = "KISS 00 Time-Lapse BLE Kybd";
//                      0123456

static long lTimeMillisecondsPhoto = 0L;
static long lTimeMillisecondsVideo = 0L;

static bool bUseNTP = true;

// Auxiliar variables to store the current output state
int bRecordingVideo = false;

static bool bReady = true;

static char sTimeBuffer[80] = "";

static char index_html[3072] = "";

// HTML web page

const char index_html_preamble[] PROGMEM = R"rawliteral(
<!DOCTYPE HTML>
<html>
<head><meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="icon" href="data:,">
<style>html { font-family: Helvetica; display: inline-block;
margin: 0px auto; text-align: center;}
.button { background-color: #4CAF50; border: none; color:
white; padding: 8px 20px;
text-decoration: none; font-size: 20px; margin: 2px; cursor:
pointer;}
.button2 {background-color: #555555;}</style>)rawliteral";

const char index_html_postamble[] PROGMEM = R"rawliteral(
<p><a href="/PHOTO"><button class="button button2">TAKE
PHOTO</button></a></p>
<p><a href="/STARTVIDEO"><button class="button button2">START
VIDEO</button></a></p>
```

```
<p><a href="/STATUSUPDATEPHOTOTIMELAPSE"><button class="button
button2">PHOTO TIME LAPSE</button></a></p>
<p><a href="/STATUSUPDATEVIDEOTIMELAPSE"><button class="button
button2">VIDEO TIME LAPSE</button></a></p>
<p><a href="/SETTINGS"><button class="button
button2">SETTINGS</button></a></p>
</body></html>)rawliteral";

const char index_html_video[] PROGMEM = R"rawliteral(
<p><a href="/STOPVIDEO"><button class="button button2">STOP
VIDEO</button></a></p>
</body></html>)rawliteral";

const char index_html_refresh[] PROGMEM = R"rawliteral(
<meta http-equiv="refresh" content="10">)rawliteral";


const char index_html_statusupdate[] PROGMEM = R"rawliteral(
<p><a href="/STOPTIMELAPSE"><button class="button button2">STOP
TIME LAPSE</button></a></p>
<p><a href="/STATUSUPDATEPHOTOTIMELAPSE"><button class="button
button2">STATUS UPDATE</button></a></p>
)rawliteral";

const char index_html_statusupdatefinished[] PROGMEM =
R"rawliteral(
<p><a href="/CONTINUE"><button class="button
button2">Continue</button></a></p>
</body></html>
)rawliteral";

const char index_html_settingspostamble[] PROGMEM =
R"rawliteral(
<form action="/GET">
<center>
<table style="border-collapse: collapse;">
  <tr>
    <td style="text-align: left;"><label>Time Delay
Minutes:</label></td>
    <td style="text-align: left;"><input type="text"
id="TIMEDELAYMINUTES"
maxlength="3" size="3" name="TIMEDELAYMINUTES"></td>
  </tr>
  <tr>
```

```
    <td style="text-align: left;"><label>Number of
Loops:</label></td>
    <td style="text-align: left;"><input type="text"
id="NUMBEROFLOOPS"
maxlength="3" size="3" name="NUMBEROFLOOPS"></td>
  </tr>
  <tr>
    <td style="text-align: left;"><label>Video Record
Seconds:</label></td>
    <td style="text-align: left;"><input type="text"
id="VIDEORECORDSECONDS"
maxlength="3" size="3" name="VIDEORECORDSECONDS"></td>
  </tr>
  <tr>
    <td style="text-align: left;"><label>Light Delay Before
Seconds:</label></td>
    <td style="text-align: left;"><input type="text"
id="LIGHTDELAYBEFORESECONDS"
maxlength="3" size="3" name="LIGHTDELAYBEFORESECONDS"></td>
  </tr>
  <tr>
    <td style="text-align: left;"><label>Light Delay After
Seconds:</label></td>
    <td style="text-align: left;"><input type="text"
id="LIGHTDELAYAFTERSECONDS"
maxlength="3" size="3" name="LIGHTDELAYAFTERSECONDS"></td>
  </tr>
  <tr>
    <td style="text-align: left;"><label>Camera
ID:</label></td>
    <td style="text-align: left;"><input type="text"
id="CAMERAID"
maxlength="3" size="3" name="CAMERAID"></td>
  </tr>
</table>
</center>
<input type="submit" value="UPDATE">
</form>
<p><a href="/CANCEL"><button class="button
button2">CANCEL</button></a></p>
<p><a href="/REBOOT"><button class="button
button2">REBOOT</button></a></p>
</body></html>
)rawliteral";
void notFound(AsyncWebServerRequest *request) {
```

```
  request->send(404, "text/plain", "Not found");
}

char sH1[] = "</head><body><h1>KISS 01 Time-Lapse<br>Camera
Controller<br> <br>";
//            01234567890123456789012345678 9
//                        1         2
void DoH1(char *sTitle)
{
  sH1[22] = cCameraID[0];
  sH1[23] = cCameraID[1];
  strcat(index_html, sH1);
  strcat(index_html, sTitle);
  strcat(index_html, "</h1>");
}

void SendOpToLoop(AsyncWebServerRequest *request, int iOp, char
*index_html)
{
  //Access KISS and perform operations

  noInterrupts();

  if (iOpCount < MAX_OPERATIONS)
  {
    iOpCount = iOpCount + 1;
    iOperations[iOpIn] = iOp;
    iOpIn = iOpIn + 1;
    if (iOpIn >= MAX_OPERATIONS)
      iOpIn = 0;
  }

  //Enable interrupts once insertion is done
  interrupts();

  request->send_P(200, "text/html", index_html);
}

void Format_index_html(bool bRoot)
{
  char sParam[10] = "";

  printLocalTime();

  strcpy(index_html, index_html_preamble);
```

```
  DoH1("");

  if (bRoot)
    strcat(index_html, VERSION);
  else
    strcat(index_html, sTimeBuffer);

  strcat(index_html, "<p>");
  strcat(index_html, "Time Delay Minutes = ");
  itoa(iTimeDelayMinutes,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Number of Loops = ");
  itoa(iNumberOfLoops,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Video Record Seconds = ");
  itoa(iVideoRecordSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Light Delay Before Seconds = ");
  itoa(iLightDelayBeforeSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Light Delay After Seconds = ");
  itoa(iLightDelayAfterSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");

  strcat(index_html, "</p>");
  strcat(index_html, index_html_postamble);
}

void ShowSettings()
{
  char sParam[10] = "";

  strcat(index_html, "Time Delay Minutes = ");
  itoa(iTimeDelayMinutes,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Number of Loops = ");
  itoa(iNumberOfLoops,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
```

```
  strcat(index_html, "Video Record Seconds = ");
  itoa(iVideoRecordSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Light Delay Before Seconds = ");
  itoa(iLightDelayBeforeSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, "Light Delay After Seconds = ");
  itoa(iLightDelayAfterSeconds,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "<br>");
  strcat(index_html, " <br>");
}

void StatusUpdate()
{
  char sParam[10] = "";

  strcpy(index_html, index_html_preamble);
  strcat(index_html, index_html_refresh);
  DoH1("TIME LAPSE");
  ShowSettings();
  strcat(index_html, "<p>");
  strcat(index_html, sTimeBuffer);
  strcat(index_html, "</p>");
  strcat(index_html, "<p>  On loop ");
  itoa(iLoopCounter,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, " of ");
  itoa(iNumberOfLoops,sParam,10);
  strcat(index_html, sParam);
  strcat(index_html, "</p>");
  strcat(index_html, index_html_statusupdate);
  strcat(index_html, "</body></html>");
}

void KISS_HTTP_Handler()
{
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    Format_index_html(true);
    request->send_P(200, "text/html", index_html);
  });
```

```
  server.on("/PHOTO", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("PHOTO");
#endif
    Format_index_html(false);
    SendOpToLoop(request, OP_PHOTO, (char *) index_html);
  });

  server.on("/STARTVIDEO", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("STARTVIDEO");
#endif
    strcpy(index_html, index_html_preamble);
    DoH1("");
    strcat(index_html, index_html_video);

    SendOpToLoop(request, OP_STARTVIDEO, (char *) index_html);
  });

  server.on("/STOPVIDEO", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("STOPVIDEO");
#endif
    Format_index_html(false);
    SendOpToLoop(request, OP_STOPVIDEO, (char *) index_html);
  });

  server.on("/STATUSUPDATEPHOTOTIMELAPSE", HTTP_GET, []
(AsyncWebServerRequest *request)
  {
#if DEBUG_OUTPUT
Serial.println("STATUSUPDATEPHOTOTIMELAPSE");
#endif
    printLocalTime();
    if (iStatusUpdateState == STATUSUPDATE_INITIAL)
    {
      StatusUpdate();
      SendOpToLoop(request, OP_STARTPHOTOTIMELAPSE, (char *)
index_html);
```

```
      iStatusUpdateState = STATUSUPDATE_RUNNING;
      return;
    }

    if (iStatusUpdateState == STATUSUPDATE_RUNNING &&
        bTimeLapseFinished == false)
    {
      StatusUpdate();
      SendOpToLoop(request, OP_STATUSUPDATE, (char *)
index_html);
      return;
    }

    iStatusUpdateState = STATUSUPDATE_FINISHED;
    iTimeLapse = NO_TIMELAPSE;
    strcpy(index_html, index_html_preamble);
    strcat(index_html, index_html_refresh);
    DoH1("TIME LAPSE FINISHED");
    strcat(index_html, index_html_statusupdatefinished);
    SendOpToLoop(request, OP_STATUSUPDATE, (char *)
index_html);
  });

  server.on("/STATUSUPDATEVIDEOTIMELAPSE", HTTP_GET, []
(AsyncWebServerRequest *request)
  {
#if DEBUG_OUTPUT
Serial.println("STATUSUPDATEVIDEOTIMELAPSE");
#endif
    printLocalTime();

    if (iStatusUpdateState == STATUSUPDATE_INITIAL)
    {
      StatusUpdate();
      SendOpToLoop(request, OP_STARTVIDEOTIMELAPSE, (char *)
index_html);
      iStatusUpdateState = STATUSUPDATE_RUNNING;
      return;
    }

    if (iStatusUpdateState == STATUSUPDATE_RUNNING &&
        bTimeLapseFinished == false)
    {
      StatusUpdate();
```

```
      SendOpToLoop(request, OP_STATUSUPDATE, (char *)
index_html);
      return;
    }

    iStatusUpdateState = STATUSUPDATE_FINISHED;
    iTimeLapse = NO_TIMELAPSE;
    strcpy(index_html, index_html_preamble);
    DoH1("TIME LAPSE FINISHED");
    strcat(index_html, index_html_statusupdatefinished);
    SendOpToLoop(request, OP_STATUSUPDATE, (char *)
index_html);
  });

  server.on("/CONTINUE", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("CONTINUE");
#endif
    iStatusUpdateState = STATUSUPDATE_INITIAL;
    Format_index_html(false);
    SendOpToLoop(request, OP_CANCEL, (char *) index_html);
  });

  server.on("/STOPTIMELAPSE", HTTP_GET, []
(AsyncWebServerRequest *request)
  {
#if DEBUG_OUTPUT
Serial.println("STOPTIMELAPSE");
#endif
    iLoopCounter = 1;
    bTimeLapseFinished = false;
    iStatusUpdateState = STATUSUPDATE_INITIAL;
    Format_index_html(false);
    SendOpToLoop(request, OP_STOPTIMELAPSE, (char *)
index_html);
  });

  server.on("/SETTINGS", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("SETTINGS");
#endif
```

```
    strcpy(index_html, index_html_preamble);
    DoH1("SETTINGS");
    ShowSettings();
    strcat(index_html, index_html_settingspostamble);
    SendOpToLoop(request, OP_SETTINGS, (char *) index_html);
  });

  server.on("/CANCEL", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("CANCEL");
#endif
    Format_index_html(false);
    SendOpToLoop(request, OP_CANCEL, (char *) index_html);
  });

  server.on("/REBOOT", HTTP_GET, [] (AsyncWebServerRequest
*request)
  {
#if DEBUG_OUTPUT
Serial.println("REBOOT");
#endif
    Format_index_html(false);
    SendOpToLoop(request, OP_REBOOT, (char *) index_html);
  });

  server.on("/GET", HTTP_GET, [] (AsyncWebServerRequest
*request) {
    String inputMessage;
    String inputParam;

#if DEBUG_OUTPUT
Serial.println("/GET");
#endif
    if (request->hasParam(TIMEDELAYMINUTES)) {
      inputMessage =
request->getParam(TIMEDELAYMINUTES)->value();
      if (inputMessage.length() > 0)
      {
        iTimeDelayMinutes = inputMessage.toInt();
        if (iTimeDelayMinutes < 1)
          iTimeDelayMinutes = 1;
      }
    }
```

```
    if (request->hasParam(NUMBEROFLOOPS)) {
      inputMessage = request->getParam(NUMBEROFLOOPS)->value();
      if (inputMessage.length() > 0)
      {
        iNumberOfLoops = inputMessage.toInt();
        if (iNumberOfLoops < 2)
          iNumberOfLoops = 2;
      }
    }

    if (request->hasParam(VIDEORECORDSECONDS)) {
      inputMessage =
request->getParam(VIDEORECORDSECONDS)->value();
      if (inputMessage.length() > 0)
      {
        iVideoRecordSeconds = inputMessage.toInt();
        if (iVideoRecordSeconds < 1)
          iVideoRecordSeconds = 1;
        lVideoRecordMilliseconds = 1000L * (long)
iVideoRecordSeconds;
      }
    }

    if (request->hasParam(LIGHTDELAYBEFORESECONDS)) {
      inputMessage =
request->getParam(LIGHTDELAYBEFORESECONDS)->value();
      if (inputMessage.length() > 0)
      {
        iLightDelayBeforeSeconds = inputMessage.toInt();
        if (iLightDelayBeforeSeconds < 1)
          iLightDelayBeforeSeconds = 1;
      }
    }

    if (request->hasParam(LIGHTDELAYAFTERSECONDS)) {
      inputMessage =
request->getParam(LIGHTDELAYAFTERSECONDS)->value();
      if (inputMessage.length() > 0)
      {
        iLightDelayAfterSeconds = inputMessage.toInt();
        if (iLightDelayAfterSeconds < 1)
          iLightDelayAfterSeconds = 1;
      }
    }
```

```
    bool bDoReboot = false;

    if (request->hasParam(CAMERAID)) {
      inputMessage = request->getParam(CAMERAID)->value();
      if (inputMessage.length() > 0)
      {
        int iCameraID = inputMessage.toInt();
        if (iCameraID > 0 && iCameraID < 100)
        {
          cCameraID[0] = (iCameraID/10) + '0';
          cCameraID[1] = (iCameraID%10) + '0';
          bDoReboot = true;
        }
      }
    }

    SetupEEPROM();

    Format_index_html(false);
    request->send(200, "text/html", index_html);

    if (bDoReboot)
      ESP.restart();

  });
}

void setup()
{
  Serial.begin(115200);
  delay(1000);

  Serial.println();
  Serial.println(PROGRAM);
  Serial.println(VERSION);
  Serial.println();

  EEPROM.begin(EEPROM_SIZE);
  if (EEPROM.read(EEPROM_SIGNATURE+0) == 'K' &&
      EEPROM.read(EEPROM_SIGNATURE+1) == 'I' &&
      EEPROM.read(EEPROM_SIGNATURE+2) == 'S' &&
      EEPROM.read(EEPROM_SIGNATURE+3) == 'S')
  {
```

```
    iNumberOfLoops =
readUnsignedIntFromEEPROM(EEPROM_NUMBER_OF_LOOPS);
    iTimeDelayMinutes =
readUnsignedIntFromEEPROM(EEPROM_TIME_DELAY_MINUTES);
    iVideoRecordSeconds =
readUnsignedIntFromEEPROM(EEPROM_VIDEO_RECORD_SECONDS);
    lVideoRecordMilliseconds = 1000L * (long)
iVideoRecordSeconds;
    iLightDelayBeforeSeconds =
readUnsignedIntFromEEPROM(EEPROM_LIGHT_DELAY_BEFORE_SECONDS);
    iLightDelayAfterSeconds =
readUnsignedIntFromEEPROM(EEPROM_LIGHT_DELAY_AFTER_SECONDS);
    cCameraID[0] = EEPROM.read(EEPROM_CAMERA_ID+0);
    cCameraID[1] = EEPROM.read(EEPROM_CAMERA_ID+1);
    if (cCameraID[0] < '0' || cCameraID[0] > '9' ||
        cCameraID[1] < '0' || cCameraID[1] > '9')
    {
      cCameraID[0] = '0';
      cCameraID[1] = '1';   // 01
    }
    sKeyboardName[5] = cCameraID[0];
    sKeyboardName[6] = cCameraID[1];
  }
  else
  {
#if DEBUG_OUTPUT
Serial.println("EEPROM KISS not found");
#endif
    SetupEEPROM();
  }
#if DEBUG_OUTPUT
Serial.println("EEPROM KISS found");
Serial.print("iTimeDelayMinutes = ");
Serial.println(iTimeDelayMinutes);
Serial.print("iNumberOfLoops = ");
Serial.println(iNumberOfLoops);
Serial.print("iVideoRecordSeconds = ");
Serial.println(iVideoRecordSeconds);
Serial.print("iLightDelayBeforeSeconds = ");
Serial.println(iLightDelayBeforeSeconds);
Serial.print("iLightDelayAfterSeconds = ");
Serial.println(iLightDelayAfterSeconds);
Serial.print("cCameraID = ");
Serial.println(cCameraID);
#endif
```

```
//  EEPROM.end();

  pinMode(LIGHT_CONTROL_PIN, OUTPUT);
  digitalWrite(LIGHT_CONTROL_PIN, LOW);

#if DEBUG_OUTPUT
  Serial.print("ESP Board MAC Address =  ");
  Serial.println(WiFi.macAddress());
  Serial.println("\n[*] Creating ESP32 AP");
#endif
  WiFi.mode(WIFI_AP_STA);

  soft_ap_ssid[5] = cCameraID[0];
  soft_ap_ssid[6] = cCameraID[1];
  WiFi.softAP(soft_ap_ssid, soft_ap_password);
  Serial.print("[+] AP Created with IP Gateway ");
  Serial.println(WiFi.softAPIP());
  Serial.println(soft_ap_ssid);

  WiFi.begin(wifi_network_ssid, wifi_network_password);
  Serial.println("\n[*] Connecting to WiFi Network");

  while(WiFi.status() != WL_CONNECTED)
  {
      Serial.print(".");
      delay(100);
  }

  Serial.print("\n[+] Connected to the WiFi network with local
IP : ");
  Serial.println(WiFi.localIP());

  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
  printLocalTime();

  KISS_HTTP_Handler();

  server.onNotFound(notFound);
  server.begin();

  // Start the Bluetooth Keyboard
  sKeyboardName[5] = cCameraID[0];
  sKeyboardName[6] = cCameraID[1];
  bleKeyboard.setName(sKeyboardName);
  bleKeyboard.begin();
```

```
  Serial.print(sKeyboardName);
  Serial.println(" -> BLE Bluetooth initiated");
}

void loop()
{
  if (iTimeLapse != NO_TIMELAPSE)
  {
    if (bRecordingVideo)
    {
      if (millis() > (lTimeMillisecondsVideo +
lVideoRecordMilliseconds))
      {
        StopVideoWithLightControl();
        if (iLoopCounter >= iNumberOfLoops)
        {
          bTimeLapseFinished = true;
        }
      }
    }

    if (millis() > (lTimeMillisecondsPhoto + (1000L * 60L *
(long) iTimeDelayMinutes)))
    {
      if (iTimeLapse == PHOTO_TIMELAPSE)
        TakePhotoWithLightControl();
      else
        StartVideoWithLightControl();

      iLoopCounter++;
      lTimeMillisecondsPhoto = millis();
      if (iLoopCounter >= iNumberOfLoops &&
          iTimeLapse == PHOTO_TIMELAPSE)
      {
        bTimeLapseFinished = true;
      }
    }
  }

  noInterrupts();

  //Access KISS and perform operations
  if (iOpCount > 0)
  {
    interrupts();
```

```
    switch (iOperations[iOpOut])
    {
      case OP_PHOTO:
#if DEBUG_OUTPUT
Serial.println("OP_PHOTO");
#endif
        TakePhotoWithLightControl();
        bRecordingVideo = false;
        break;

      case OP_STARTVIDEO:
#if DEBUG_OUTPUT
Serial.println("OP_STARTVIDEO");
#endif
        digitalWrite(LIGHT_CONTROL_PIN, HIGH);
        delay(iLightDelayBeforeSeconds * 1000L);
        bleKeyboard.write(KEY_MEDIA_VOLUME_UP);
        bRecordingVideo = true;
        break;

      case OP_STOPVIDEO:
#if DEBUG_OUTPUT
Serial.println("OP_STOPVIDEO");
#endif
        bRecordingVideo = false;
        bleKeyboard.write(KEY_MEDIA_VOLUME_DOWN);
        delay(iLightDelayAfterSeconds * 1000L);
        digitalWrite(LIGHT_CONTROL_PIN, LOW);
        break;

      case OP_STARTPHOTOTIMELAPSE:
#if DEBUG_OUTPUT
Serial.println("OP_STARTPHOTOTIMELAPSE");
#endif
        TakePhotoWithLightControl();
        if (iNumberOfLoops > 1)
        {
          iTimeLapse = PHOTO_TIMELAPSE;
          lTimeMillisecondsPhoto = millis();
          iLoopCounter = 1;
        }
        break;

      case OP_STARTVIDEOTIMELAPSE:
#if DEBUG_OUTPUT
```

```
Serial.println("OP_STARTVIDEOTIMELAPSE");
#endif
        StartVideoWithLightControl();
        if (iNumberOfLoops > 1)
        {
          iTimeLapse = VIDEO_TIMELAPSE;
          lTimeMillisecondsPhoto = millis();
          iLoopCounter = 1;
        }
        break;

      case OP_STOPTIMELAPSE:
#if DEBUG_OUTPUT
Serial.println("OP_STOPTIMELAPSE");
#endif
        iLoopCounter = 0;
        //if (iTimeLapse == VIDEO_TIMELAPSE)
        if (bRecordingVideo)
        {
          StopVideoWithLightControl();
        }
        iTimeLapse = NO_TIMELAPSE;
        bTimeLapseFinished = false;
        iStatusUpdateState = STATUSUPDATE_INITIAL;
        break;

      case OP_SETTINGS:
#if DEBUG_OUTPUT
Serial.println("OP_SETTINGS");
#endif
        break;

      case OP_CANCEL:
#if DEBUG_OUTPUT
Serial.println("OP_CANCEL");
#endif
        bTimeLapseFinished = false;
        break;

      case OP_REBOOT:
#if DEBUG_OUTPUT
Serial.println("OP_REBOOT");
Serial.println("REBOOT KISS Time-Lapse 01 Camera Controller");
#endif
        ESP.restart();
```

```
        break;

      case OP_STATUSUPDATE:
#if DEBUG_OUTPUT
Serial.println("OP_STATUSUPDATE");
#endif
        break;

    }
    iOpCount = iOpCount - 1;
    iOpOut = iOpOut + 1;
    if (iOpOut >=  MAX_OPERATIONS)
      iOpOut = 0;
  }
  else
  {
    //Unlock once operations are done
    interrupts();
  }
}

void printLocalTime()
{
  if (bUseNTP == false)
  {
    sTimeBuffer[0] = '\0';
    return;
  }

  struct tm timeinfo;
  if(!getLocalTime(&timeinfo))
  {
    bUseNTP = false;
#if DEBUG_OUTPUT
    Serial.println("Failed to obtain time");
#endif
    return;
  }
  strftime(sTimeBuffer,sizeof(sTimeBuffer),"%Y-%m-%d %H:%M:%S",
&timeinfo);
#if DEBUG_OUTPUT
  Serial.println(sTimeBuffer);
#endif
}
```

```
void TakePhotoWithLightControl()
{
#if DEBUG_OUTPUT
Serial.println("TAKE PHOTO");
#endif

  digitalWrite(LIGHT_CONTROL_PIN, HIGH);
  delay(iLightDelayBeforeSeconds * 1000L);

  bleKeyboard.write(KEY_MEDIA_VOLUME_UP);

  delay(iLightDelayAfterSeconds * 1000L);
  digitalWrite(LIGHT_CONTROL_PIN, LOW);
}

void StartVideoWithLightControl()
{
#if DEBUG_OUTPUT
Serial.println("START VIDEO");
#endif

  digitalWrite(LIGHT_CONTROL_PIN, HIGH);
  delay(iLightDelayBeforeSeconds * 1000L);

  bleKeyboard.write(KEY_MEDIA_VOLUME_UP);

  lTimeMillisecondsVideo = millis();
  bRecordingVideo = true;
}

void StopVideoWithLightControl()
{
#if DEBUG_OUTPUT
Serial.println("StopVideoWithLightControl");
#endif

  bleKeyboard.write(KEY_MEDIA_VOLUME_DOWN);

  bRecordingVideo = false;
  delay(iLightDelayAfterSeconds * 1000L);
  digitalWrite(LIGHT_CONTROL_PIN, LOW);
}

void writeUnsignedIntIntoEEPROM(int address, unsigned int
number)
```

```
{
  EEPROM.write(address, number >> 8);
  EEPROM.write(address + 1, number & 0xFF);
}

unsigned int readUnsignedIntFromEEPROM(int address)
{
  return (EEPROM.read(address) << 8) + EEPROM.read(address +
1);
}

void SetupEEPROM()
{
  EEPROM.begin(100);
  EEPROM.write(EEPROM_SIGNATURE+0, 'K');
  EEPROM.write(EEPROM_SIGNATURE+1, 'I');
  EEPROM.write(EEPROM_SIGNATURE+2, 'S');
  EEPROM.write(EEPROM_SIGNATURE+3, 'S');
  writeUnsignedIntIntoEEPROM(EEPROM_NUMBER_OF_LOOPS,
iNumberOfLoops);
  writeUnsignedIntIntoEEPROM(EEPROM_TIME_DELAY_MINUTES,
iTimeDelayMinutes);
  writeUnsignedIntIntoEEPROM(EEPROM_VIDEO_RECORD_SECONDS,
iVideoRecordSeconds);
  writeUnsignedIntIntoEEPROM(EEPROM_LIGHT_DELAY_BEFORE_SECONDS,
iLightDelayBeforeSeconds);
  writeUnsignedIntIntoEEPROM(EEPROM_LIGHT_DELAY_AFTER_SECONDS,
iLightDelayAfterSeconds);
  EEPROM.write(EEPROM_CAMERA_ID+0, cCameraID[0]);
  EEPROM.write(EEPROM_CAMERA_ID+1, cCameraID[1]);
  EEPROM.commit();

#if DEBUG_OUTPUT
Serial.println("EEPROM KISS initialized");
#endif
}
```

KISS Image Processing and Video Creation

Installation and Setup for Video Creation

Install Python

https://www.python.org/downloads/

Install Image Magick

https://imagemagick.org/script/download.php

Install ffmpeg

https://ffmpeg.org/download.html

Detailed Mac Instructions

Install Homebrew

xcode-select --install

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Install Python

https://www.python.org/downloads/

Install Image Magick

brew install imagemagick

Install ffmpeg

brew install ffmpeg

Install pip

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python3 get-pip.py

Install exifread

pip3 install exifread

Install scipy

pip3 install scipy

Install VideoCreation Python Program

Create folder

Documents/VideoCreation

Download VideoCreation.py

https://github.com/TomRolander/VideoCreation/blob/main/VideoCreation.py

Use Copy Option

Use TextEdit to save VideoCreation.py

Launch TextEdit
Preferences -> Plain Text
Copy VideoCreation.py contents into TextEdit document
Save as VideoCreation.py in the Documents/VideoCreation folder

Verify execution of VideoCreation.py

python3 VideoCreation.py –version

VideoCreation

VideoCreation Python Program Operation

```
C:\Users\rolander\Documents\VideoCreation>py VideoCreation.py
--help
VideoCreation Ver 0.2 2022-11-02
usage: Video Creation from Photos
      [-h]
      [--verbose]
      [--inputdir INPUTDIR]
      [--outputdir OUTPUTDIR]
      [--nextimagenumber NEXTIMAGENUMBER]
      [--fps FPS]
      [--crf CRF]
      [--numberofpoints NUMBEROFPOINTS]
      [--videocreate {True,False}]

options:
  -h, --help            show this help message and exit
  --verbose             Show more context
  --inputdir INPUTDIR
  --outputdir OUTPUTDIR
  --nextimagenumber NEXTIMAGENUMBER
  --fps FPS
  --crf CRF             CRF range 0-50 default is 23
  --numberofpoints NUMBEROFPOINTS
  --videocreate {True,False}
```

```
"""
/*************************************************************************
**
  VideoCreation
    The purpose of this Python program is to input a folder of image
  files (JPG), watermark the images (ImageMagick), rename the files,
  and create a video of the watermarked files (FFMPEG).

  Original Code:  2022-10-08

  Tom Rolander, MSEE
  Mentor, Circuit Design, Software, and 3D Printing
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu


*************************************************************************
*/

 launch.json additional values

                "--inputdir", ".\\Input",
                "--outputdir", ".\\Output",
                "--fps", "16",

To Do List
    - compute average time between pics
    - add optional string to -annotate
            -annotate 90x90+150+30 %[exif:DateTimeOriginal]_fps" +
str(frames_per_second).zfill(2) + " -fill

"""
Version = "Ver 0.3"
RevisionDate = "2023-03-03"

import sys
import os
import time
import subprocess
import argparse
import time
import exifread
import datetime
import array
import statistics
```

```
import platform

from scipy import stats

from datetime import datetime
from datetime import timedelta

NoneType = type(None)

minutes_elapsed = array.array('f')

separator = "\\"

print ("VideoCreation",Version, RevisionDate)
if platform.system() == "Windows":
    print("Running on Windows")
else:
    print("Running on Mac")
    separator = "/"

parser = argparse.ArgumentParser("Video Creation from Photos")
parser.add_argument('--verbose', action='store_true', help="Show more
context")
parser.add_argument('--inputdir', type=str, required=False)
parser.add_argument('--outputdir', type=str, required=False)
parser.add_argument('--nextimagenumber', type=int, required=False)
parser.add_argument('--fps', type=int, required=False)
parser.add_argument('--crf', type=int, required=False, help="CRF range
0-50 default is 23")
parser.add_argument('--numberofpoints', type=int, required=False)
parser.add_argument('--videocreate', required=False,
choices=('True','False'))
args = parser.parse_args()

if args.verbose:
    print(f"Create MP4 time lapse video from a folder of JPGs")

now = datetime.now()
start = time.time()

start_time = now.strftime("%H:%M:%S")
print("Start Time =", start_time)

if type(args.inputdir) is NoneType:
    input_dir_name = os.getcwd() + separator + "Input"
else:
    input_dir_name = args.inputdir
print("Input Dir = ", end ="")
print(input_dir_name)
```

```
if type(args.outputdir) is NoneType:
    output_dir_name = os.getcwd() + separator + "Output"
    watermarked_dir_name = output_dir_name + separator + "Watermarked"
    videos_dir_name = output_dir_name + separator + "Videos"
else:
    output_dir_name = args.outputdir
    watermarked_dir_name = args.outputdir + separator + "Watermarked"
    videos_dir_name = args.outputdir + separator + "Videos"
print("Output Dir Watermarked = ", end ="")
print(watermarked_dir_name)
print("Output Dir Videos = ", end ="")
print(videos_dir_name)

if type(args.nextimagenumber) is NoneType:
    isExist = os.path.exists(watermarked_dir_name + separator + "0")
    if (isExist == False):
        base = 0
    else:
        base = 0
        # Iterate directory
        for path in os.listdir(watermarked_dir_name + separator + "0"):
            # check if current path is a file
            if os.path.isfile(os.path.join(watermarked_dir_name +
separator + "0", path)):
                base += 1
else:
    base = args.nextimagenumber - 1
print("Next image number = ", end ="")
print(base+1)

if type(args.fps) is NoneType:
    frames_per_second = 16
else:
    frames_per_second = args.fps
print("Frames per second = ", end ="")
print(frames_per_second)

if type(args.crf) is NoneType:
    constant_rate_factor = 23
else:
    constant_rate_factor = args.crf
print("Constant rate factor = ", end ="")
print(constant_rate_factor)

if type(args.numberofpoints) is NoneType:
    number_of_points = 10
else:
    number_of_points = args.numberofpoints
```

```
print("Number of points = ", end ="")
print(number_of_points)

isExist = os.path.exists(input_dir_name)
if (isExist == False):
    print("Input Dir does not exist", input_dir_name)
    exit(1)

count = 0

# Iterate directory

# Get the file count
for path in os.listdir(input_dir_name):
    # check if current path is a file
    #print(path)
    if (path.endswith('JPG') == False) and (path.endswith('jpg') ==
False) and (path.endswith('jpeg') == False):
        continue
    if os.path.isfile(os.path.join(input_dir_name, path)):
        count += 1
        #print(os.path.join(input_dir_name, path))
print('Input file count:', count)

if count % number_of_points != 0:
    print("Your number of input files is not a multiple of your number of
points!")
    exit(1)

havepreviousvalue = False
index = 0
nmb = 0

print('Rename files to EXIF Date Taken')
for path in os.listdir(input_dir_name):
    # check if current path is a file
    #print(path)
    if (path.endswith('JPG') == False) and (path.endswith('jpg') ==
False) and (path.endswith('jpeg') == False):
        continue
    with open(os.path.join(input_dir_name, path), "rb") as image:
        exif = exifread.process_file(image)
        dt = str(exif['EXIF DateTimeOriginal']) #get 'Date Taken' from
JPG
        ds = time.strptime(dt, '%Y:%m:%d %H:%M:%S')

        if ((number_of_points == 1) or ((index % number_of_points) ==
0)):
            dd = datetime.strptime(dt, '%Y:%m:%d %H:%M:%S')
```

```
            if havepreviousvalue == False:
                havepreviousvalue = True
            else:
                delta = dd - dd_previous
                minutes_elapsed.append(delta.total_seconds()/60)
                #print(f"Time difference is {minutes_elapsed[nmb]}
minutes")
                nmb = nmb + 1
            dd_previous = dd
        index = index + 1

        nt = time.strftime("%Y-%m-%d_%H-%M-%S",ds)
        newname = nt + ".JPG"
        image.close()
        #print("Rename " + os.path.join(input_dir_name,path) + " to " +
os.path.join(input_dir_name,newname))
        os.rename(os.path.join(input_dir_name,path),
os.path.join(input_dir_name,newname))

if (index > 0):
    print("Mean minutes between photos = ", end ="")
    print("{:.1f}".format(statistics.mean(minutes_elapsed)))
    print("Trim mean minutes between photos of 0.025 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.025)))
    print("Trim mean minutes between photos of 0.05 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.05)))
    print("Trim mean minutes between photos of 0.10 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.10)))
    print("Trim mean minutes between photos of 0.15 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.15)))
    print("Trim mean minutes between photos of 0.20 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.20)))
    print("Trim mean minutes between photos of 0.25 = ", end ="")
    print("{:.1f}".format(stats.trim_mean(minutes_elapsed, 0.25)))
    hours_per_second_of_video = "{:.1f}".format((frames_per_second *
stats.trim_mean(minutes_elapsed, 0.10) ) / 60)
else:
    hours_per_second_of_video = "0.0"

print("1 Sec Video = ", end ="")
print(hours_per_second_of_video, end ="")
print(" Hour Real Time")
hours_per_second_of_video = "1Sec" + hours_per_second_of_video + "Hr"

if type(args.videocreate) is NoneType or args.videocreate == True:
    print("Create MP4 video")
    bVideocreate = True
else:
    print("Only watermark files, no video will be created")
```

```
    bVideocreate = False

isExist = os.path.exists(output_dir_name)
if (isExist == False):
    print("Creating Output Dir ", output_dir_name)
    os.mkdir(output_dir_name)

isExist = os.path.exists(watermarked_dir_name)
if (isExist == False):
    print("Creating Output Watermarked Dir ", watermarked_dir_name)
    os.mkdir(watermarked_dir_name)

if bVideocreate == True:
    isExist = os.path.exists(videos_dir_name)
    if (isExist == False):
        print("Creating Output Videos Dir ", videos_dir_name)
        os.mkdir(videos_dir_name)

# Get list of all files only in the given directory
##list_of_files = filter( lambda x:
os.path.isfile(os.path.join(input_dir_name, x)),
##                        os.listdir(input_dir_name) )
# Sort list of files based on last modification time in ascending order
##list_of_files = sorted( list_of_files,
##                        key = lambda x:
os.path.getmtime(os.path.join(input_dir_name, x))
##                        )

list_of_files = sorted( filter( lambda x:
os.path.isfile(os.path.join(input_dir_name, x)),
                       os.listdir(input_dir_name) ) )

for i in range(0, number_of_points):
    #print("\"" + watermarked_dir_name + separator + "" + str(i) + "\"")
    isExist = os.path.exists(watermarked_dir_name + separator + "" +
str(i))
    if (isExist == False):
        print("\"" + watermarked_dir_name + separator + "" + str(i) +
"\"")
        os.mkdir(watermarked_dir_name + separator + "" + str(i))

index = 0
#base = 0
print("Watermarking photos:")
for file_name in list_of_files:
    if ((index % number_of_points) == 0):
        base += 1
    input_file_path = os.path.join(input_dir_name, file_name)
    #timestamp_str = time.strftime(  '%m/%d/%Y %H:%M:%S',
```

```
    #
time.gmtime(os.path.getmtime(input_file_path)))
    print(str(index+1).zfill(6), str((index % number_of_points)),
str(base).zfill(6), file_name)
    output_file_path =  watermarked_dir_name + separator + "" +
str((index % number_of_points)) + separator + "" + str(base).zfill(6) +
".jpg"
#    output_file_path =  watermarked_dir_name + separator + "" +
str((index % number_of_points)) + separator + "" + file_name
    #print(output_file_path)

    os.system("magick convert " + "\"" + input_file_path + "\"" + "
-quiet -gravity northwest -font Arial-bold -pointsize 144 -fill black
-annotate 90x90+150+30 %[exif:DateTimeOriginal] -fill white -annotate
90x90+155+35 %[exif:DateTimeOriginal] \"" + output_file_path + "\"")
    #print ("Done")
    index += 1

if bVideocreate == True:
    for i in range(0, number_of_points):
        #print (output_dir_name + separator + "" + "P" + str(i) +
"_%05d")
        #print (videos_dir_name + separator + "" + "VideoPoint_" + str(i)
+ ".mp4")
        print ("ffmpeg -loglevel error -r " + str(frames_per_second) + "
-f image2 -s 1920x1080 -i " + "\"" + watermarked_dir_name + separator +
"" + str(i) + separator + ""+ "%06d.jpg" + "\"" + " -vcodec libx264 -crf
" + str(constant_rate_factor) + " -pix_fmt yuv420p -y " + "\"" +
videos_dir_name + separator + "" + "Pt" + str(i) + "_fps" +
str(frames_per_second).zfill(2) + "_crf" +
str(constant_rate_factor).zfill(2) + "_" + hours_per_second_of_video +
".mp4" + "\"")
        os.system("ffmpeg -loglevel error -r " + str(frames_per_second) +
" -f image2 -s 1920x1080 -i " + "\"" + watermarked_dir_name + separator +
"" + str(i) + separator + ""+ "%06d.jpg" + "\"" + " -vcodec libx264 -crf
" + str(constant_rate_factor) + " -pix_fmt yuv420p -y " + "\"" +
videos_dir_name + separator + "" + "Pt" + str(i) + "_fps" +
str(frames_per_second).zfill(2) + "_crf" +
str(constant_rate_factor).zfill(2) + "_" + hours_per_second_of_video +
".mp4" + "\"")

end_time = now.strftime("%H:%M:%S")
print("End Time =", end_time)
end = time.time()
print("Elapsed time = ", end ="")
#print('{:.1f}'.format(end - start))
td = timedelta(seconds=int(end - start))
print(td)
```