JULY 24, 2014

M1-D MULTISENSOR IMAGING PLATFORM

CALEB KOCH

Table of Contents

Hardware configuration:

The IR camera has three wires: power, video out, and rs485. Only the power and rs485 need to be connected for use with m1-D camera. Video out is optional.

First connect the rs485 cable to the rs485 shield. The black wire (negative) should be plugged into the port closest to the numbers: 22-23-2021. Refer to *Figure 1*.

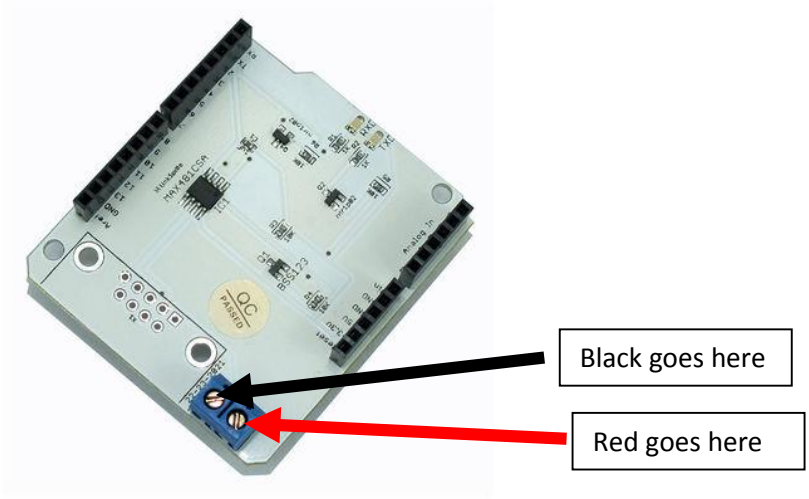


Figure 1

Setup the power cable. Once, connected the camera should do a self-test routine where it pans 360° and simultaneously tilts up and down.

When the program is ready to be uploaded, remove the rs485 shield then click the upload button in the Arduino IDE. Once the upload bar disappears, the shield can be replaced. *It is important to note that every time a new program is uploaded, the shield must be removed.*

To configure the ps/2 keyboard, make sure the ends of the wires are spliced and soldered so they can fit into the headers on the Arduino. Connect the white cable to ground, the red to 5V, the yellow to pin 3, and the green to pin 2. (The wire without insulation is not used).

Communicating with Camera

The camera can only accept commands using the Pelco D protocol which has the standard syntax as:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sync	Camera Address	Command 1	Command 2	Data 1	Data 2	Checksum

Each byte will be a hexadecimal number which should be preceded by 0x in Arduino.

The sync byte is always set to FF.

The camera address will be 1 for first camera, 2 for second camera, etc…

Data 1 is the pan speed which can be set from 10 to 3F and FF for turbo speed.

Data 2 is the tilt speed and can be set in the same way as pan speed.

The checksum is the sum of bytes 2 through 6.

Command 1 and 2 are formed using the following table:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Command 1	Sense	Reserved	Reserved	Auto / Manual Scan	Camera On/Off	Iris Close	Iris Open	Focus Near
Command 2	Focus Far	Zoom Wide	Zoom Tele	Tilt Down	Tilt Up	Pan Left	Pan Right	Fixed to 0

Each bit is either 1 (on) or 0 (off). When the bits are set, a binary number will remain which can then be converted to hexadecimal. (For example pan left – 0 0 0 0 0 1 0 0 – converts to 04 in hex).

The sense bit indicates the meaning of bits 4 and 3. If the sense bit is set to 1, and bits 4 and 3 are set to 1, the command will enable auto-scan and turn the camera on. If the sense bit is off and bits 4 and 3 are on the command will enable manual scan and turn the camera off. Furthermore, if either bit 4 or bit 3 are off then no action will be taken for those features.

The Reserved bits are set to 0.

Zoom wide means zoom out and zoom tele means zoom in.

Refer to Mapping Keys section for information on the meaning of certain terminology (ex. Iris open, focus near, etc...).

If using an xbox 360 controller to generate commands, then a separate process is needed. The method I used involved converting input from controller to keystrokes on the computer, mapping the keystrokes to a message, recognizing the message, and executing a Pelco d command based on the message. While the method works, it is slow and leaves a lot of room for error, but I will include it for modification purposes.

First, a software is needed to convert input from controller to keystrokes. There is plenty of freeware to choose from, but I used JoyToKey, which can be found here.

A processing program is then used to recognize the keystroke and convert it to an integer which is then outputted along the usb cable connected to the Arduino. The code itself can be found in

the programming section. When running the program, a window is generated which must be selected or else the keystrokes will not be recognized.

The Arduino picks up the integer and executes the appropriate pelco d command. This program can also be found in the programming section.

Click here for a link to download the processing IDE.

Important Pelco D commands for use with M1-D infrared camera are listed below (0x means hexadecimal format. Bytes are separated by commas).

- Pan left fast: 0xFF,0x01,0x00,0x04,0x3F,0x00,0x44
- Pan right fast: 0xFF,0x01,0x00,0x02,0x3F,0x00,0x42
- Tilt up fast: 0xFF,0x01,0x00,0x08,0x00,0x3F,0x48
- Tilt down fast: 0xFF,0x01,0x00,0x10,0x00,0x3F,0x50
- Zoom in fast: 0xFF,0x01,0x00,0x20,0x00,0x03,0x24
- Zoom out fast: 0xFF,0x01,0x00,0x40,0x00,0x03,0x44
- Change color pallete: 0xFF,0x01,0x04,0x00,0x00,0x00,0x05
- Activate laser pointer: 0xFF,0x01,0x00,0x80,0x00,0x00,0x81
- Switch visible/thermal: 0xFF,0x01,0x01,0x00,0x00,0x00,0x02
- Pan left slow: 0xFF,0x01,0x00,0x04,0x10,0x00,0x15
- Pan right slow: 0xFF,0x01,0x00,0x02,0x10,0x00,0x13
- Zoom in slow: 0xFF,0x01,0x00,0x20,0x00,0x01,0x22
- Zoom out slow: 0xFF,0x01,0x00,0x40,0x00,0x01,0x42
- Iris open: 0xFF,0x01,0x02,0x00,0x00,0x00,0x03
- Iris close: 0xFF,0x01,0x04,0x00,0x00,0x00,0x05
- Access menu: 0xFF,0x01,0x00,0x07,0x00,0x63,0x6b
- Go to preset: 0xFF,0x01,0x00,0x07,0x00,0x01 to 0xFF,varies

Below is a chart listing the supported presets. The number needs to be converted from decimal to hexadecicmal and then placed in the 6th byte of the "go to preset" command.

#	Function	Desc
81	Video Switch	Switch video between thermal and visual
82	Auto Scan	Scan from presets 0~39 sequentially
84	Pattern Scan 1	Pattern scan from presets 40~49
85	Pattern Scan 2	Pattern scan from presets 50~59
86	Pattern Scan 3	Pattern scan from presets 60~69
87	Pattern Scan 4	Pattern scan from presets 70~79
89	Clear General Presets	Clear all the position settings of all general presets
90	OSD On/Off	Turn on/off On Screen Display (logos etc.)
94	Remote Reboot	Reboot The M1-D
96	180 Degree Continuous Scan	Continuously scan the area 90 degrees to the right and left of the start position. (+/- 90Degree)
97	360 Degree Random Scan	Pan randomly in 360 Degrees
98	90 Degree Continuous Scan	Continuously scan the area 45 degrees to the right and left of the start position. (+/- 45Degree)
99	Call OSD	Call the On Screen Display Menu

Programming

When programming the Arduino to execute Pelco D commands, there are a few things to keep in mind.

The process of interfacing with the M1-D camera involves serial communication. Thus, in the setup function of the Arduino sketch, the baud rate needs to be initialized using the syntax: Serial.begin(2400) where 2400 is the bps. Important: the baud rate is always set to 2400.

When sending the hexadecimal code to the camera, the Serial.write(*byte*) command is used where *byte* is the hexadecimal number.

Once a command is executed, the camera will respond until told to do otherwise (e.g. if the camera is told to pan left it will do so perpetually). To avoid extraneous movement, make sure to send a stop command after the normal command.

When running a program, the camera will output ASCII characters onto the Serial Monitor thus rendering the serial buffer inaccessible. This implys that any type of input cannot come via the Serial Monitor which is why various other types of devices are used (xbox controller and ps/2 keyboard).

When programming in processing, it is important to declare the serial port being used by the arduino. To find the name of the COM port, one can open the arduino IDE, it should be listed in the bottom right hand corner. Alternatively, the COM port can be found in device manager under *Ports (COM & LPT).*

Below is the program used to interface with the camera using the ps/2 keyboard. Make sure to download the PS2Keyboard library before running the program. Click here for the website.

```
#include <PS2Keyboard.h>

const int DataPin = 3; //define where the data cable (yellow) is connected
const int IRQpin = 2; //define where the clock cable (green) is connected

int i=0; //variable for setting speed
byte c; //stores incoming byte
int counter=0; //variable for managing loops

//Store each command as an array of type byte. Each can be executed using a simple loop
byte lf[7]={0xFF,0x01,0x00,0x04,0x3F,0x00,0x44};//left pan fast
byte rf[7]={0xFF,0x01,0x00,0x02,0x3F,0x00,0x42};//right pan fast
byte tdf[7]={0xFF,0x01,0x00,0x08,0x00,0xFF,0x108};//tilt down fast
byte tuf[7]={0xFF,0x01,0x00,0x10,0x00,0xFF,0x110};//tilt up fast
byte vts[7]={0xFF,0x01,0x01,0x00,0x00,0x00,0x02};//switch visible/thermal
```

```
byte ccp[7]={0xFF,0x01,0x04,0x00,0x00,0x00,0x05};//change color pallete
byte zos[7]={0xFF,0x01,0x00,0x40,0x00,0x01,0x42};//zoom out slow
byte zis[7]={0xFF,0x01,0x00,0x20,0x00,0x01,0x22};//zoom in slow
byte cs[7]={0xFF,0x01,0x00,0x07,0x00,0x63,0x6b}; //access menu
byte alp[7]= {0xFF,0x01,0x00,0x80,0x00,0x00,0x81};//activate laser pointer
byte ic[7]= {0xFF,0x01,0x04,0x00,0x00,0x00,0x05};//iris close
byte io[7]= {0xFF,0x01,0x02,0x00,0x00,0x00,0x03};//iris open
byte s[7]={0xFF,0x01,0x00,0x00,0x00,0x00,0x01};//stop
byte cs180[7]={0xFF,0x01,0x00,0x07,0x00,0x60,0x68};//continuous scan +/- 90 degrees from start position
byte rs360[7]={0xFF,0x01,0x00,0x07,0x00,0x61,0x69};//random scan 360 degrees
byte cs90[7]={0xFF,0x01,0x00,0x07,0x00,0x62,0x6a};//continuous scan +/- 45 degrees from start position

PS2Keyboard keyboard; //initialize the keyboard

void setup()
{
  Serial.begin(2400); //set the baud rate to 2400 always
  Serial.println("begin"); //test to make sure the serial moniter is working properly
  keyboard.begin(DataPin, IRQpin); //initialize the data and interrupt pins
}

void loop()
{
    if(keyboard.available()) //if there is a byte in the serial buffer the statement will be set to true and conditional will
be executed
    {
      c=keyboard.read(); //take the first byte and store it in variable c
    }
    //first conditional does noting
    if(c=='0')
    {
    }

    /*
    if '1' is pressed on the keyboard set the variable to a small number.
    Later in the program this will be the delay. A larger delay allows the command to be
    executed for a longer period of time, thus allowing the camera to reach a higher speed. A
    smaller delay corresponds to the opposite.
    */
    else if(c=='1')
    {
      i=10;
    }

    else if(c=='2')
    {
     i=50;
    }

    else if(c=='3')
    {
     i=100;
    }

    else if (c==PS2_UPARROW)
    {
```

```
    //execute each element of the array one at a time, for all seven elements
     while(counter<7)
    {
      Serial.write(tuf[counter]);
      counter+=1;
    }
    delay(i); //delay allows the camera to have time to respond
    counter=0; // reset the counter
    stopcmmnd(); //call the stopcmmnd() function which tells the camera to stop. Without this, the camera would countinue tilting.
    c=0x00; //reset the byte
   }

   else if(c==PS2_DOWNARROW)
   {
     while(counter<7)
    {
      Serial.write(tdf[counter]);
      counter+=1;
    }
    delay(i);
    counter=0;
    stopcmmnd();
    c=0x00;
   }

   else if(c==PS2_RIGHTARROW)
   {
     while(counter<7)
    {
      Serial.write(rf[counter]);
      counter+=1;
    }
    delay(i);
    counter=0;
    c=0x00;
    stopcmmnd();
   }

   else if(c==PS2_LEFTARROW)
   {
     while(counter<7)
    {
      Serial.write(lf[counter]);
      counter+=1;
    }
    delay(i);
    counter=0;
    c=0x00;
    stopcmmnd();
   }

   else if(c==PS2_ENTER)
   {
     while(counter<7)
    {
```

```
    Serial.write(io[counter]);
    counter+=1;
   }
  counter=0;
  c=0x00;

 }
 else if(c==PS2_BACKSPACE)
 {
   while(counter<7)
  {
    Serial.write(ic[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;

 }

 else if(c=='c')
 {
   while(counter<7)
  {
    Serial.write(ccp[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;

 }

 else if(c== PS2_ESC)
 {
   while(counter<7)
  {
    Serial.write(s[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;

 }

 else if(c=='m')
 {
   while(counter<7)
  {
    Serial.write(cs[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;
  stopcmmnd();
  delay(i);
 }
```

```
else if(c=='z')
{
  while(counter<7)
  {
    Serial.write(zis[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;
  stopcmmnd();
  delay(i);
}

else if(c=='x')
{
  while(counter<7)
  {
    Serial.write(zos[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;
  stopcmmnd();
  delay(i);
}

else if(c=='s')
{
  while(counter<7)
  {
    Serial.write(vts[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;
  stopcmmnd();
  delay(i);
}

else if(c=='q')
{
  while(counter<7)
  {
    Serial.write(cs180[counter]);
    counter+=1;
  }
  counter=0;
  c=0x00;

  delay(i);
}
else if(c=='w')
 {
  while(counter<7)
  {
    Serial.write(rs360[counter]);
```

```
      counter+=1;
     }
     counter=0;
     c=0x00;

     delay(i);
   }
   else if(c=='e')
    {
     while(counter<7)
     {
      Serial.write(cs90[counter]);
      counter+=1;
     }
     counter=0;
     c=0x00;
     delay(i);
   }

   /*if there is any problem, comment out this section (just the else conditional, not the stopcmmnd function) and use the esc key when necessary
   else*/
   {
     while(counter<7)
     {
      Serial.write(s[counter]);
      counter+=1;
     }
     counter=0;
   }

}

void stopcmmnd()
{
  while(counter<7)
     {
      Serial.write(s[counter]);
      counter+=1;
     }
     counter=0;
}
```

Here is the processing program used to convert keystrokes to an integer.

```
import processing.serial.*;
Serial port;
void setup()
{
  size(256, 150);

  port = new Serial(this, "COM7", 2400); //user may need to change the COM port based on where the arduino is connected
}

void draw()
{
  if (keyPressed && key == 'a')
  {
    int msg = 0;
    port.write(msg);
    println("a has been pressed");
  }
  else if (keyPressed && key == 'w')
  {
    int msg = 1;
    port.write(msg);
    println("w has been pressed");
  }
  else if (keyPressed && key == 'd')
  {
    int msg = 2;
    port.write(msg);
  }
  else if (keyPressed && key == 's')
  {
    int msg = 3;
    port.write(msg);
  }
    else if (keyPressed && key == 'e')
    {
    int msg = 4;
    port.write(msg);
  }
    else if (keyPressed && key == 'z')
    {
    int msg = 5;
    port.write(msg);
  }
    else if (keyPressed && key == 'x')
    {
    int msg = 6;
    port.write(msg);
  }
    else if (keyPressed && key == 'c')
    {
    int msg = 7;
    port.write(msg);
  }
```

```
    else if (keyPressed && key == 'v')
    {
  int msg = 8;
  port.write(msg);
 }
    else if (keyPressed && key == 'b')
    {
  int msg = 9;
  port.write(msg);
 }
    else if (keyPressed && key == 'n')
    {
  int msg = 10;
  port.write(msg);
 }
    else if (keyPressed && key == 'm')
    {
  int msg = 11;
  port.write(msg);
 }
    else if (keyPressed && key == ',')
    {
  int msg = 12;
  port.write(msg);
 }
 else if (keyPressed && key == 'u')
    {
  int msg = 13;
  port.write(msg);
 }


 else
 {
  int msg = 50;
  port.write(msg);
 }
}
```

Here is the program used to recognize the message sent by the processing program and send the appropriate Pelco d command to the M1-D camera.

```
#define DATA 2
#define WR   3
#define CS   4
#define CS2  5

const int ledPin = 13; //optional. I used a LED for debugging purposes

int counter=0;
byte lf[7]={0xFF,0x01,0x00,0x04,0x3F,0x00,0x44};//left pan fast
byte rf[7]={0xFF,0x01,0x00,0x02,0x3F,0x00,0x42};//right pan fast
byte tuf[7]={0xFF,0x01,0x00,0x08,0x00,0x3F,0x48};//tilt up fast
byte tdf[7]={0xFF,0x01,0x00,0x10,0x00,0x3F,0x50};//tilt down fast
byte vts[7]={0xFF,0x01,0x01,0x00,0x00,0x00,0x02};//switch visible/thermal
byte ccp[7]={0xFF,0x01,0x04,0x00,0x00,0x00,0x05};//change color pallete
byte zos[7]={0xFF,0x01,0x00,0x40,0x00,0x01,0x42};//zoom out slow
byte zis[7]={0xFF,0x01,0x00,0x20,0x00,0x01,0x22};//zoom in slow
byte cs[7]={0xFF,0x01,0x00,0x07,0x00,0x63,0x6b}; //access menu
byte alp[7]= {0xFF,0x01,0x00,0x80,0x00,0x00,0x81};//activate laser pointer
byte ic[7]= {0xFF,0x01,0x04,0x00,0x00,0x00,0x05};//iris close
byte io[7]= {0xFF,0x01,0x02,0x00,0x00,0x00,0x03};//iris open
byte s[7]={0xFF,0x01,0x00,0x00,0x00,0x00,0x01};//stop
byte cs180[7]={0xFF,0x01,0x00,0x07,0x00,0x60,0x68};//continuous scan +/- 90 degrees from start position
byte rs360[7]={0xFF,0x01,0x00,0x07,0x00,0x61,0x69};//random scan 360 degrees
byte cs90[7]={0xFF,0x01,0x00,0x07,0x00,0x62,0x6a};//continuous scan +/- 45 degrees from start position


void setup()
  {
    Serial.begin(2400);
    pinMode(ledPin, OUTPUT);
    digitalWrite(ledPin, HIGH);
    delay(100);
    digitalWrite(ledPin, LOW);
  }

void loop()
  {

  // check if data has been sent from the computer
  if(Serial.available())
  {
    byte proc;
    // read the most recent byte
    proc = Serial.read();


    if (proc == 0)
    {
      digitalWrite(ledPin, HIGH);
      while(counter<7)
      {
        Serial.write(lf[counter]);
```

```
    counter+=1;
   }
   delay(50); //allow time for camera to respond
   proc=0x00; //reset the proc
   counter=0; //reset the counter so it can be used again
 }

  else if  (proc == 1)
   {
    digitalWrite(ledPin, HIGH);
    while(counter<7)
     {
      Serial.write(tuf[counter]);
      counter+=1;
     }
    delay(50);
    proc=0x00;
   counter=0;
  }

else if(proc==2)
{
  digitalWrite(ledPin, HIGH);
  while(counter<7)
    {
     Serial.write(rf[counter]);
     counter+=1;
    }
   delay(50);
   proc=0x00;
   counter=0;
}
else if(proc==3)
{
  digitalWrite(ledPin, HIGH);
  while(counter<7)
    {
     Serial.write(tdf[counter]);
     counter+=1;
    }
   delay(50);
   proc=0x00;
   counter=0;
}

  else if  (proc == 4)
   {
    digitalWrite(ledPin, HIGH);
   while(counter<7)
    {
     Serial.write(vts[counter]);
     counter+=1;
    }
   delay(50);
   proc=0x00;
   counter=0;
```

```
}
else if  (proc == 5)
{
 digitalWrite(ledPin, HIGH);
 while(counter<7)
  {
    Serial.write(io[counter]);
    counter+=1;
  }
  delay(50);
  proc=0x00;
  counter=0;
}
else if  (proc == 6)
{
 digitalWrite(ledPin, HIGH);
 while(counter<7)
  {
    Serial.write(ic[counter]);
    counter+=1;
  }
  delay(50);
  proc=0x00;
  counter=0;
}
else if  (proc == 7)
{
  digitalWrite(ledPin, HIGH);
 while(counter<7)
  {
    Serial.write(zis[counter]);
    counter+=1;
  }
  delay(50);
  proc=0x00;
  counter=0;
}
else if  (proc == 8)
{
 digitalWrite(ledPin, HIGH);
 while(counter<7)
  {
    Serial.write(zos[counter]);
    counter+=1;
  }
  delay(50);
  proc=0x00;
  counter=0;
}

else if  (proc == 9)
{
 digitalWrite(ledPin, HIGH);
 while(counter<7)
  {
    Serial.write(ccp[counter]);
```

```
    counter+=1;
   }
   delay(50);
   proc=0x00;
   counter=0;
 }
 else if  (proc == 10)
 {
  digitalWrite(ledPin, HIGH);
  while(counter<7)
   {
    Serial.write(cs180[counter]);
    counter+=1;
   }
   delay(50);
   proc=0x00;
   counter=0;
 }
 else if  (proc == 11)
 {
  digitalWrite(ledPin, HIGH);
  while(counter<7)
   {
    Serial.write(rs360[counter]);
    counter+=1;
   }
   delay(50);
   proc=0x00;
   counter=0;
 }
 else if  (proc == 12)
 {
  digitalWrite(ledPin, HIGH);
  while(counter<7)
   {
    Serial.write(cs90[counter]);
    counter+=1;
   }
   delay(50);
   proc=0x00;
   counter=0;
 }
 else if  (proc == 13)
 {
  digitalWrite(ledPin, HIGH);
  while(counter<7)
   {
    Serial.write(cs[counter]);
    counter+=1;
   }
   delay(50);
   proc=0x00;
   counter=0;
 }

 else
```

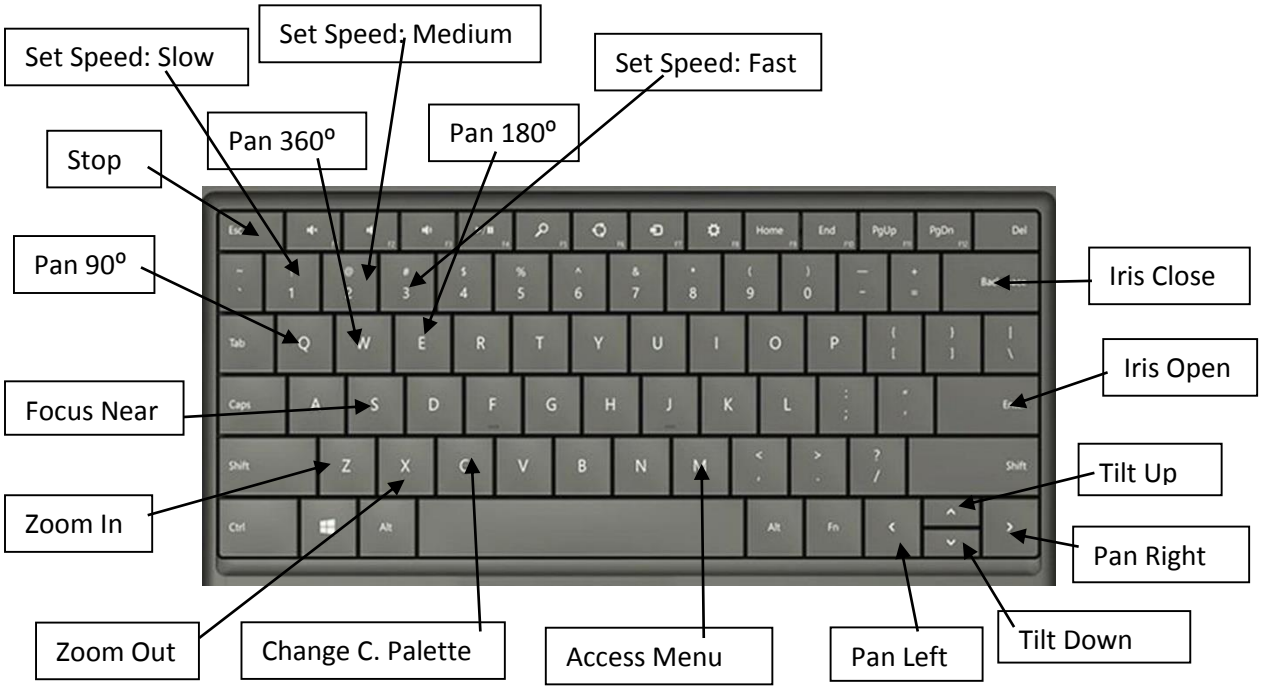
```
    {
      digitalWrite(ledPin, LOW);
      while(counter<7)
      {
        Serial.write(s[counter]);
        counter+=1;
      }
      counter=0;
    }

}

  }
```

Mapping Keys

Use the following diagram as reference when using the ps/2 keyboard to control the M1-D camera.



Iris open is used to control thermal imaging sensor zoom and is also used as select when navigating the menu.

Iris close is used to change color palettes and as a ‘back’ button when navigating the menu.

Focus Near is used to switch between visible and thermal imaging.

Focus Far is not used here but activates the laser pointer.

This diagram shows the command generated with each corresponding button when using the xbox controller.

Important to note: if using the JoyToKey software, button 1 is the equivalent of 'A' on the xbox controller, button 2 is 'B', button 3 is 'Y', button 4 is 'X', and button 8 is Start.

Here is a chart showing the complete mapping of each key. The proc # is the integer messge sent by the processing program. The keyboard key is the keystroke generated by the xbox controller.

proc #	Command	Keyboard Key	Xbox Input
0	Left Fast	a	Stick 1 left
1	Tilt up Fast	w	Stick 1 up
2	Right Fast	d	Stick 1 right
3	Tilt down Fast	s	Stick 1 down
4	Change color pallete	e	X
5	Iris Open	z	A
6	Iris Close	x	B
7	Zoom in	c	Stick 2 up
8	Zoom out	v	Stick 2 down
9	Switch Visual/Thermal	b	Y
10	Continuous scan 180	n	d-pad up
11	Random scan 360	m	d-pad right
12	Continuous scan 90	,	d-pad left
13	Onscreen Display	u	Start

**Charts for commands taken from:
http://www.commfront.com/RS232_Examples/CCTV/Pelco_D_Pelco_P_Examples_Tutorial.HTM
**Title picture and preset table taken from M1-D Manual