//This code is to use with dual axis fpv camera cradle that uses two servos, adding a joystick module

//We control the position of the cradle by moving the analog stick, the cradle move to the direction where the stick is pointing

//until the limits and stay there

//Refer to surtrtech.com for more information

#include <Servo.h> //Servos library and declaration

Servo myservo1;

Servo myservo2;

int a,b,X,Y; //Variables needed later

int YAxis = 1; //Declaring where the X axis and Y axis of the joystick pins are wired

int XAxis = 0; //Of course analog inputs

void setup() {

Serial.begin(9600); //Setting the Serial monitor baude rate and launching

pinMode(XAxis, INPUT); //Declaring the pin modes and servo pins

myservo1.attach(8);

pinMode(YAxis, INPUT);

myservo2.attach(9);

}

void loop() {

a=myservo1.read(); //Reading the previous servos positions is an important step so we can know where they should position next

b=myservo2.read();

X=analogRead(XAxis);//Reading the joystick values

Y=analogRead(YAxis);

if(X>550){ //Here we didn't do any calibration so the joystick has three positions (Left|Resting|Right)

a=a-1; //it depends on the value we read we can know in which direction the stick is pointing and I left the resting position big actually it's just 1 value

myservo1.write(a); //we inject the new value

delay(50); //You can make the delay big or short or act on a=a-x to make big steps or short steps

}

if(X<450){

a=a+1; //Here we did the opposit operation to move to the opposit direction

myservo1.write(a);

delay(50);

}

if(Y>600){ //Here we didn't do any calibration so the joystick has three positions (Up|Resting|Down) ditto

b=b+1;

myservo2.write(b);

delay(50);

}

if(Y<450){

b=b-1;

myservo2.write(b);

delay(50);

}

// Code for streaming video to screen  
// https://circuitdigest.com/microcontroller-projects/how-to-use-ov7670-camera-module-with-arduino

#include <stdint.h>

#include <avr/io.h>

#include <util/twi.h>

#include <util/delay.h>

#include <avr/pgmspace.h>

#define F\_CPU 16000000UL

#define vga   0

#define qvga  1

#define qqvga   2

#define yuv422  0

#define rgb565  1

#define bayerRGB  2

#define camAddr\_WR  0x42

#define camAddr\_RD  0x43

/\* Registers \*/

#define REG\_GAIN    0x00  /\* Gain lower 8 bits (rest in vref) \*/

#define REG\_BLUE    0x01  /\* blue gain \*/

#define REG\_RED       0x02  /\* red gain \*/

#define REG\_VREF    0x03  /\* Pieces of GAIN, VSTART, VSTOP \*/

#define REG\_COM1    0x04  /\* Control 1 \*/

#define COM1\_CCIR656  0x40    /\* CCIR656 enable \*/

#define REG\_BAVE    0x05  /\* U/B Average level \*/

#define REG\_GbAVE   0x06  /\* Y/Gb Average level \*/

#define REG\_AECHH   0x07  /\* AEC MS 5 bits \*/

#define REG\_RAVE    0x08  /\* V/R Average level \*/

#define REG\_COM2    0x09  /\* Control 2 \*/

#define COM2\_SSLEEP         0x10  /\* Soft sleep mode \*/

#define REG\_PID           0x0a  /\* Product ID MSB \*/

#define REG\_VER           0x0b  /\* Product ID LSB \*/

#define REG\_COM3    0x0c  /\* Control 3 \*/

#define COM3\_SWAP         0x40  /\* Byte swap \*/

#define COM3\_SCALEEN          0x08  /\* Enable scaling \*/

#define COM3\_DCWEN          0x04  /\* Enable downsamp/crop/window \*/

#define REG\_COM4    0x0d  /\* Control 4 \*/

#define REG\_COM5    0x0e  /\* All "reserved" \*/

#define REG\_COM6    0x0f  /\* Control 6 \*/

#define REG\_AECH    0x10  /\* More bits of AEC value \*/

#define REG\_CLKRC   0x11  /\* Clocl control \*/

#define CLK\_EXT           0x40  /\* Use external clock directly \*/

#define CLK\_SCALE   0x3f  /\* Mask for internal clock scale \*/

#define REG\_COM7    0x12  /\* Control 7 \*/ //REG mean address.

#define COM7\_RESET          0x80  /\* Register reset \*/

#define COM7\_FMT\_MASK         0x38

#define COM7\_FMT\_VGA          0x00

#define COM7\_FMT\_CIF          0x20  /\* CIF format \*/

#define COM7\_FMT\_QVGA         0x10  /\* QVGA format \*/

#define COM7\_FMT\_QCIF         0x08  /\* QCIF format \*/

#define COM7\_RGB          0x04  /\* bits 0 and 2 - RGB format \*/

#define COM7\_YUV          0x00  /\* YUV \*/

#define COM7\_BAYER          0x01  /\* Bayer format \*/

#define COM7\_PBAYER         0x05  /\* "Processed bayer" \*/

#define REG\_COM8    0x13  /\* Control 8 \*/

#define COM8\_FASTAEC          0x80  /\* Enable fast AGC/AEC \*/

#define COM8\_AECSTEP          0x40  /\* Unlimited AEC step size \*/

#define COM8\_BFILT    0x20  /\* Band filter enable \*/

#define COM8\_AGC    0x04  /\* Auto gain enable \*/

#define COM8\_AWB    0x02  /\* White balance enable \*/

#define COM8\_AEC    0x01  /\* Auto exposure enable \*/

#define REG\_COM9    0x14  /\* Control 9- gain ceiling \*/

#define REG\_COM10   0x15  /\* Control 10 \*/

#define COM10\_HSYNC         0x40  /\* HSYNC instead of HREF \*/

#define COM10\_PCLK\_HB         0x20  /\* Suppress PCLK on horiz blank \*/

#define COM10\_HREF\_REV          0x08  /\* Reverse HREF \*/

#define COM10\_VS\_LEAD         0x04  /\* VSYNC on clock leading edge \*/

#define COM10\_VS\_NEG          0x02  /\* VSYNC negative \*/

#define COM10\_HS\_NEG          0x01  /\* HSYNC negative \*/

#define REG\_HSTART    0x17  /\* Horiz start high bits \*/

#define REG\_HSTOP   0x18  /\* Horiz stop high bits \*/

#define REG\_VSTART    0x19  /\* Vert start high bits \*/

#define REG\_VSTOP   0x1a  /\* Vert stop high bits \*/

#define REG\_PSHFT   0x1b  /\* Pixel delay after HREF \*/

#define REG\_MIDH    0x1c  /\* Manuf. ID high \*/

#define REG\_MIDL    0x1d  /\* Manuf. ID low \*/

#define REG\_MVFP    0x1e  /\* Mirror / vflip \*/

#define MVFP\_MIRROR         0x20  /\* Mirror image \*/

#define MVFP\_FLIP   0x10  /\* Vertical flip \*/

#define REG\_AEW           0x24  /\* AGC upper limit \*/

#define REG\_AEB           0x25    /\* AGC lower limit \*/

#define REG\_VPT           0x26  /\* AGC/AEC fast mode op region \*/

#define REG\_HSYST   0x30  /\* HSYNC rising edge delay \*/

#define REG\_HSYEN   0x31  /\* HSYNC falling edge delay \*/

#define REG\_HREF    0x32  /\* HREF pieces \*/

#define REG\_TSLB    0x3a  /\* lots of stuff \*/

#define TSLB\_YLAST    0x04  /\* UYVY or VYUY - see com13 \*/

#define REG\_COM11   0x3b  /\* Control 11 \*/

#define COM11\_NIGHT         0x80  /\* NIght mode enable \*/

#define COM11\_NMFR          0x60  /\* Two bit NM frame rate \*/

#define COM11\_HZAUTO          0x10  /\* Auto detect 50/60 Hz \*/

#define COM11\_50HZ          0x08  /\* Manual 50Hz select \*/

#define COM11\_EXP   0x02

#define REG\_COM12   0x3c  /\* Control 12 \*/

#define COM12\_HREF          0x80  /\* HREF always \*/

#define REG\_COM13   0x3d  /\* Control 13 \*/

#define COM13\_GAMMA         0x80  /\* Gamma enable \*/

#define COM13\_UVSAT         0x40  /\* UV saturation auto adjustment \*/

#define COM13\_UVSWAP          0x01  /\* V before U - w/TSLB \*/

#define REG\_COM14   0x3e  /\* Control 14 \*/

#define COM14\_DCWEN         0x10  /\* DCW/PCLK-scale enable \*/

#define REG\_EDGE    0x3f  /\* Edge enhancement factor \*/

#define REG\_COM15   0x40  /\* Control 15 \*/

#define COM15\_R10F0         0x00  /\* Data range 10 to F0 \*/

#define COM15\_R01FE         0x80  /\*      01 to FE \*/

#define COM15\_R00FF         0xc0  /\*      00 to FF \*/

#define COM15\_RGB565          0x10  /\* RGB565 output \*/

#define COM15\_RGB555          0x30  /\* RGB555 output \*/

#define REG\_COM16   0x41  /\* Control 16 \*/

#define COM16\_AWBGAIN         0x08  /\* AWB gain enable \*/

#define REG\_COM17   0x42  /\* Control 17 \*/

#define COM17\_AECWIN          0xc0  /\* AEC window - must match COM4 \*/

#define COM17\_CBAR          0x08  /\* DSP Color bar \*/

/\*

\* This matrix defines how the colors are generated, must be

\* tweaked to adjust hue and saturation.

\*

\* Order: v-red, v-green, v-blue, u-red, u-green, u-blue

\* They are nine-bit signed quantities, with the sign bit

\* stored in0x58.Sign for v-red is bit 0, and up from there.

\*/

#define REG\_CMATRIX\_BASE  0x4f

#define CMATRIX\_LEN           6

#define REG\_CMATRIX\_SIGN  0x58

#define REG\_BRIGHT    0x55  /\* Brightness \*/

#define REG\_CONTRAS         0x56  /\* Contrast control \*/

#define REG\_GFIX    0x69  /\* Fix gain control \*/

#define REG\_REG76   0x76  /\* OV's name \*/

#define R76\_BLKPCOR         0x80  /\* Black pixel correction enable \*/

#define R76\_WHTPCOR         0x40  /\* White pixel correction enable \*/

#define REG\_RGB444          0x8c  /\* RGB 444 control \*/

#define R444\_ENABLE         0x02  /\* Turn on RGB444, overrides 5x5 \*/

#define R444\_RGBX   0x01  /\* Empty nibble at end \*/

#define REG\_HAECC1    0x9f  /\* Hist AEC/AGC control 1 \*/

#define REG\_HAECC2    0xa0  /\* Hist AEC/AGC control 2 \*/

#define REG\_BD50MAX         0xa5  /\* 50hz banding step limit \*/

#define REG\_HAECC3    0xa6  /\* Hist AEC/AGC control 3 \*/

#define REG\_HAECC4    0xa7  /\* Hist AEC/AGC control 4 \*/

#define REG\_HAECC5    0xa8  /\* Hist AEC/AGC control 5 \*/

#define REG\_HAECC6    0xa9  /\* Hist AEC/AGC control 6 \*/

#define REG\_HAECC7    0xaa  /\* Hist AEC/AGC control 7 \*/

#define REG\_BD60MAX         0xab  /\* 60hz banding step limit \*/

#define REG\_GAIN    0x00  /\* Gain lower 8 bits (rest in vref) \*/

#define REG\_BLUE    0x01  /\* blue gain \*/

#define REG\_RED           0x02  /\* red gain \*/

#define REG\_VREF    0x03  /\* Pieces of GAIN, VSTART, VSTOP \*/

#define REG\_COM1    0x04  /\* Control 1 \*/

#define COM1\_CCIR656          0x40  /\* CCIR656 enable \*/

#define REG\_BAVE    0x05  /\* U/B Average level \*/

#define REG\_GbAVE   0x06  /\* Y/Gb Average level \*/

#define REG\_AECHH   0x07  /\* AEC MS 5 bits \*/

#define REG\_RAVE    0x08  /\* V/R Average level \*/

#define REG\_COM2    0x09  /\* Control 2 \*/

#define COM2\_SSLEEP         0x10  /\* Soft sleep mode \*/

#define REG\_PID           0x0a  /\* Product ID MSB \*/

#define REG\_VER           0x0b  /\* Product ID LSB \*/

#define REG\_COM3    0x0c  /\* Control 3 \*/

#define COM3\_SWAP         0x40  /\* Byte swap \*/

#define COM3\_SCALEEN          0x08  /\* Enable scaling \*/

#define COM3\_DCWEN          0x04  /\* Enable downsamp/crop/window \*/

#define REG\_COM4    0x0d  /\* Control 4 \*/

#define REG\_COM5    0x0e  /\* All "reserved" \*/

#define REG\_COM6    0x0f  /\* Control 6 \*/

#define REG\_AECH    0x10  /\* More bits of AEC value \*/

#define REG\_CLKRC   0x11  /\* Clocl control \*/

#define CLK\_EXT           0x40  /\* Use external clock directly \*/

#define CLK\_SCALE   0x3f  /\* Mask for internal clock scale \*/

#define REG\_COM7    0x12  /\* Control 7 \*/

#define COM7\_RESET          0x80  /\* Register reset \*/

#define COM7\_FMT\_MASK         0x38

#define COM7\_FMT\_VGA          0x00

#define COM7\_FMT\_CIF          0x20  /\* CIF format \*/

#define COM7\_FMT\_QVGA         0x10  /\* QVGA format \*/

#define COM7\_FMT\_QCIF         0x08  /\* QCIF format \*/

#define COM7\_RGB    0x04  /\* bits 0 and 2 - RGB format \*/

#define COM7\_YUV    0x00  /\* YUV \*/

#define COM7\_BAYER          0x01  /\* Bayer format \*/

#define COM7\_PBAYER         0x05  /\* "Processed bayer" \*/

#define REG\_COM8    0x13  /\* Control 8 \*/

#define COM8\_FASTAEC          0x80  /\* Enable fast AGC/AEC \*/

#define COM8\_AECSTEP          0x40  /\* Unlimited AEC step size \*/

#define COM8\_BFILT    0x20  /\* Band filter enable \*/

#define COM8\_AGC    0x04  /\* Auto gain enable \*/

#define COM8\_AWB    0x02  /\* White balance enable \*/

#define COM8\_AEC    0x01  /\* Auto exposure enable \*/

#define REG\_COM9    0x14  /\* Control 9- gain ceiling \*/

#define REG\_COM10   0x15  /\* Control 10 \*/

#define COM10\_HSYNC         0x40  /\* HSYNC instead of HREF \*/

#define COM10\_PCLK\_HB         0x20  /\* Suppress PCLK on horiz blank \*/

#define COM10\_HREF\_REV          0x08  /\* Reverse HREF \*/

#define COM10\_VS\_LEAD           0x04  /\* VSYNC on clock leading edge \*/

#define COM10\_VS\_NEG          0x02  /\* VSYNC negative \*/

#define COM10\_HS\_NEG          0x01  /\* HSYNC negative \*/

#define REG\_HSTART    0x17  /\* Horiz start high bits \*/

#define REG\_HSTOP   0x18  /\* Horiz stop high bits \*/

#define REG\_VSTART    0x19  /\* Vert start high bits \*/

#define REG\_VSTOP   0x1a  /\* Vert stop high bits \*/

#define REG\_PSHFT   0x1b  /\* Pixel delay after HREF \*/

#define REG\_MIDH    0x1c  /\* Manuf. ID high \*/

#define REG\_MIDL    0x1d  /\* Manuf. ID low \*/

#define REG\_MVFP    0x1e  /\* Mirror / vflip \*/

#define MVFP\_MIRROR         0x20  /\* Mirror image \*/

#define MVFP\_FLIP   0x10  /\* Vertical flip \*/

#define REG\_AEW           0x24  /\* AGC upper limit \*/

#define REG\_AEB           0x25  /\* AGC lower limit \*/

#define REG\_VPT           0x26  /\* AGC/AEC fast mode op region \*/

#define REG\_HSYST   0x30  /\* HSYNC rising edge delay \*/

#define REG\_HSYEN   0x31  /\* HSYNC falling edge delay \*/

#define REG\_HREF    0x32  /\* HREF pieces \*/

#define REG\_TSLB    0x3a  /\* lots of stuff \*/

#define TSLB\_YLAST    0x04  /\* UYVY or VYUY - see com13 \*/

#define REG\_COM11   0x3b  /\* Control 11 \*/

#define COM11\_NIGHT         0x80  /\* NIght mode enable \*/

#define COM11\_NMFR          0x60  /\* Two bit NM frame rate \*/

#define COM11\_HZAUTO          0x10  /\* Auto detect 50/60 Hz \*/

#define COM11\_50HZ          0x08  /\* Manual 50Hz select \*/

#define COM11\_EXP   0x02

#define REG\_COM12   0x3c  /\* Control 12 \*/

#define COM12\_HREF          0x80  /\* HREF always \*/

#define REG\_COM13   0x3d  /\* Control 13 \*/

#define COM13\_GAMMA         0x80  /\* Gamma enable \*/

#define COM13\_UVSAT         0x40  /\* UV saturation auto adjustment \*/

#define COM13\_UVSWAP          0x01  /\* V before U - w/TSLB \*/

#define REG\_COM14   0x3e  /\* Control 14 \*/

#define COM14\_DCWEN         0x10  /\* DCW/PCLK-scale enable \*/

#define REG\_EDGE    0x3f  /\* Edge enhancement factor \*/

#define REG\_COM15   0x40  /\* Control 15 \*/

#define COM15\_R10F0         0x00  /\* Data range 10 to F0 \*/

#define COM15\_R01FE         0x80  /\*      01 to FE \*/

#define COM15\_R00FF         0xc0  /\*      00 to FF \*/

#define COM15\_RGB565          0x10  /\* RGB565 output \*/

#define COM15\_RGB555          0x30  /\* RGB555 output \*/

#define REG\_COM16   0x41  /\* Control 16 \*/

#define COM16\_AWBGAIN         0x08  /\* AWB gain enable \*/

#define REG\_COM17   0x42  /\* Control 17 \*/

#define COM17\_AECWIN          0xc0  /\* AEC window - must match COM4 \*/

#define COM17\_CBAR          0x08  /\* DSP Color bar \*/

#define CMATRIX\_LEN             6

#define REG\_BRIGHT    0x55  /\* Brightness \*/

#define REG\_REG76   0x76  /\* OV's name \*/

#define R76\_BLKPCOR         0x80  /\* Black pixel correction enable \*/

#define R76\_WHTPCOR         0x40  /\* White pixel correction enable \*/

#define REG\_RGB444          0x8c  /\* RGB 444 control \*/

#define R444\_ENABLE         0x02  /\* Turn on RGB444, overrides 5x5 \*/

#define R444\_RGBX   0x01  /\* Empty nibble at end \*/

#define REG\_HAECC1    0x9f  /\* Hist AEC/AGC control 1 \*/

#define REG\_HAECC2    0xa0  /\* Hist AEC/AGC control 2 \*/

#define REG\_BD50MAX         0xa5  /\* 50hz banding step limit \*/

#define REG\_HAECC3    0xa6  /\* Hist AEC/AGC control 3 \*/

#define REG\_HAECC4    0xa7  /\* Hist AEC/AGC control 4 \*/

#define REG\_HAECC5    0xa8  /\* Hist AEC/AGC control 5 \*/

#define REG\_HAECC6    0xa9  /\* Hist AEC/AGC control 6 \*/

#define REG\_HAECC7    0xaa  /\* Hist AEC/AGC control 7 \*/

#define REG\_BD60MAX         0xab  /\* 60hz banding step limit \*/

#define MTX1            0x4f  /\* Matrix Coefficient 1 \*/

#define MTX2            0x50  /\* Matrix Coefficient 2 \*/

#define MTX3            0x51  /\* Matrix Coefficient 3 \*/

#define MTX4            0x52  /\* Matrix Coefficient 4 \*/

#define MTX5            0x53  /\* Matrix Coefficient 5 \*/

#define MTX6            0x54  /\* Matrix Coefficient 6 \*/

#define REG\_CONTRAS         0x56  /\* Contrast control \*/

#define MTXS            0x58  /\* Matrix Coefficient Sign \*/

#define AWBC7           0x59  /\* AWB Control 7 \*/

#define AWBC8           0x5a  /\* AWB Control 8 \*/

#define AWBC9           0x5b  /\* AWB Control 9 \*/

#define AWBC10            0x5c  /\* AWB Control 10 \*/

#define AWBC11            0x5d  /\* AWB Control 11 \*/

#define AWBC12            0x5e  /\* AWB Control 12 \*/

#define REG\_GFI           0x69  /\* Fix gain control \*/

#define GGAIN           0x6a  /\* G Channel AWB Gain \*/

#define DBLV            0x6b  

#define AWBCTR3           0x6c  /\* AWB Control 3 \*/

#define AWBCTR2           0x6d  /\* AWB Control 2 \*/

#define AWBCTR1           0x6e  /\* AWB Control 1 \*/

#define AWBCTR0           0x6f  /\* AWB Control 0 \*/

struct regval\_list{

  uint8\_t reg\_num;

  uint16\_t value;

};

const struct regval\_list qvga\_ov7670[] PROGMEM = {

  { REG\_COM14, 0x19 },

  { 0x72, 0x11 },

  { 0x73, 0xf1 },

  { REG\_HSTART, 0x16 },

  { REG\_HSTOP, 0x04 },

  { REG\_HREF, 0xa4 },

  { REG\_VSTART, 0x02 },

  { REG\_VSTOP, 0x7a },

  { REG\_VREF, 0x0a },

  { 0xff, 0xff }, /\* END MARKER \*/

};

const struct regval\_list yuv422\_ov7670[] PROGMEM = {

  { REG\_COM7, 0x0 },  /\* Selects YUV mode \*/

  { REG\_RGB444, 0 },  /\* No RGB444 please \*/

  { REG\_COM1, 0 },

  { REG\_COM15, COM15\_R00FF },

  { REG\_COM9, 0x6A }, /\* 128x gain ceiling; 0x8 is reserved bit \*/

  { 0x4f, 0x80 },   /\* "matrix coefficient 1" \*/

  { 0x50, 0x80 },   /\* "matrix coefficient 2" \*/

  { 0x51, 0 },    /\* vb \*/

  { 0x52, 0x22 },   /\* "matrix coefficient 4" \*/

  { 0x53, 0x5e },   /\* "matrix coefficient 5" \*/

  { 0x54, 0x80 },   /\* "matrix coefficient 6" \*/

  { REG\_COM13, COM13\_UVSAT },

  { 0xff, 0xff },   /\* END MARKER \*/

};

const struct regval\_list ov7670\_default\_regs[] PROGMEM = {//from the linux driver

  { REG\_COM7, COM7\_RESET },

  { REG\_TSLB, 0x04 }, /\* OV \*/

  { REG\_COM7, 0 },  /\* VGA \*/

  /\*

  \* Set the hardware window.  These values from OV don't entirely

  \* make sense - hstop is less than hstart.  But they work...

  \*/

  { REG\_HSTART, 0x13 }, { REG\_HSTOP, 0x01 },

  { REG\_HREF, 0xb6 }, { REG\_VSTART, 0x02 },

  { REG\_VSTOP, 0x7a }, { REG\_VREF, 0x0a },

  { REG\_COM3, 0 }, { REG\_COM14, 0 },

  /\* Mystery scaling numbers \*/

  { 0x70, 0x3a }, { 0x71, 0x35 },

  { 0x72, 0x11 }, { 0x73, 0xf0 },

  { 0xa2,/\* 0x02 changed to 1\*/1 }, { REG\_COM10, 0x0 },

  /\* Gamma curve values \*/

  { 0x7a, 0x20 }, { 0x7b, 0x10 },

  { 0x7c, 0x1e }, { 0x7d, 0x35 },

  { 0x7e, 0x5a }, { 0x7f, 0x69 },

  { 0x80, 0x76 }, { 0x81, 0x80 },

  { 0x82, 0x88 }, { 0x83, 0x8f },

  { 0x84, 0x96 }, { 0x85, 0xa3 },

  { 0x86, 0xaf }, { 0x87, 0xc4 },

  { 0x88, 0xd7 }, { 0x89, 0xe8 },

  /\* AGC and AEC parameters.  Note we start by disabling those features,

  then turn them only after tweaking the values. \*/

  { REG\_COM8, COM8\_FASTAEC | COM8\_AECSTEP },

  { REG\_GAIN, 0 }, { REG\_AECH, 0 },

  { REG\_COM4, 0x40 }, /\* magic reserved bit \*/

  { REG\_COM9, 0x18 }, /\* 4x gain + magic rsvd bit \*/

  { REG\_BD50MAX, 0x05 }, { REG\_BD60MAX, 0x07 },

  { REG\_AEW, 0x95 }, { REG\_AEB, 0x33 },

  { REG\_VPT, 0xe3 }, { REG\_HAECC1, 0x78 },

  { REG\_HAECC2, 0x68 }, { 0xa1, 0x03 }, /\* magic \*/

  { REG\_HAECC3, 0xd8 }, { REG\_HAECC4, 0xd8 },

  { REG\_HAECC5, 0xf0 }, { REG\_HAECC6, 0x90 },

  { REG\_HAECC7, 0x94 },

  { REG\_COM8, COM8\_FASTAEC | COM8\_AECSTEP | COM8\_AGC | COM8\_AEC },

  { 0x30, 0 }, { 0x31, 0 },//disable some delays

  /\* Almost all of these are magic "reserved" values.  \*/

  { REG\_COM5, 0x61 }, { REG\_COM6, 0x4b },

  { 0x16, 0x02 }, { REG\_MVFP, 0x07 },

  { 0x21, 0x02 }, { 0x22, 0x91 },

  { 0x29, 0x07 }, { 0x33, 0x0b },

  { 0x35, 0x0b }, { 0x37, 0x1d },

  { 0x38, 0x71 }, { 0x39, 0x2a },

  { REG\_COM12, 0x78 }, { 0x4d, 0x40 },

  { 0x4e, 0x20 }, { REG\_GFIX, 0 },

  /\*{0x6b, 0x4a},\*/{ 0x74, 0x10 },

  { 0x8d, 0x4f }, { 0x8e, 0 },

  { 0x8f, 0 }, { 0x90, 0 },

  { 0x91, 0 }, { 0x96, 0 },

  { 0x9a, 0 }, { 0xb0, 0x84 },

  { 0xb1, 0x0c }, { 0xb2, 0x0e },

  { 0xb3, 0x82 }, { 0xb8, 0x0a },

  /\* More reserved magic, some of which tweaks white balance \*/

  { 0x43, 0x0a }, { 0x44, 0xf0 },

  { 0x45, 0x34 }, { 0x46, 0x58 },

  { 0x47, 0x28 }, { 0x48, 0x3a },

  { 0x59, 0x88 }, { 0x5a, 0x88 },

  { 0x5b, 0x44 }, { 0x5c, 0x67 },

  { 0x5d, 0x49 }, { 0x5e, 0x0e },

  { 0x6c, 0x0a }, { 0x6d, 0x55 },

  { 0x6e, 0x11 }, { 0x6f, 0x9e }, /\* it was 0x9F "9e for advance AWB" \*/

  { 0x6a, 0x40 }, { REG\_BLUE, 0x40 },

  { REG\_RED, 0x60 },

  { REG\_COM8, COM8\_FASTAEC | COM8\_AECSTEP | COM8\_AGC | COM8\_AEC | COM8\_AWB },

  /\* Matrix coefficients \*/

  { 0x4f, 0x80 }, { 0x50, 0x80 },

  { 0x51, 0 },    { 0x52, 0x22 },

  { 0x53, 0x5e }, { 0x54, 0x80 },

  { 0x58, 0x9e },

  { REG\_COM16, COM16\_AWBGAIN }, { REG\_EDGE, 0 },

  { 0x75, 0x05 }, { REG\_REG76, 0xe1 },

  { 0x4c, 0 },     { 0x77, 0x01 },

  { REG\_COM13, /\*0xc3\*/0x48 }, { 0x4b, 0x09 },

  { 0xc9, 0x60 },   /\*{REG\_COM16, 0x38},\*/

  { 0x56, 0x40 },

  { 0x34, 0x11 }, { REG\_COM11, COM11\_EXP | COM11\_HZAUTO },

  { 0xa4, 0x82/\*Was 0x88\*/ }, { 0x96, 0 },

  { 0x97, 0x30 }, { 0x98, 0x20 },

  { 0x99, 0x30 }, { 0x9a, 0x84 },

  { 0x9b, 0x29 }, { 0x9c, 0x03 },

  { 0x9d, 0x4c }, { 0x9e, 0x3f },

  { 0x78, 0x04 },

  /\* Extra-weird stuff.  Some sort of multiplexor register \*/

  { 0x79, 0x01 }, { 0xc8, 0xf0 },

  { 0x79, 0x0f }, { 0xc8, 0x00 },

  { 0x79, 0x10 }, { 0xc8, 0x7e },

  { 0x79, 0x0a }, { 0xc8, 0x80 },

  { 0x79, 0x0b }, { 0xc8, 0x01 },

  { 0x79, 0x0c }, { 0xc8, 0x0f },

  { 0x79, 0x0d }, { 0xc8, 0x20 },

  { 0x79, 0x09 }, { 0xc8, 0x80 },

  { 0x79, 0x02 }, { 0xc8, 0xc0 },

  { 0x79, 0x03 }, { 0xc8, 0x40 },

  { 0x79, 0x05 }, { 0xc8, 0x30 },

  { 0x79, 0x26 },

  { 0xff, 0xff }, /\* END MARKER \*/

};

void error\_led(void){

  DDRB |= 32;//make sure led is output

  while (1){//wait for reset

    PORTB ^= 32;// toggle led

    \_delay\_ms(100);

  }

}

void twiStart(void){

  TWCR = \_BV(TWINT) | \_BV(TWSTA) | \_BV(TWEN);//send start

  while (!(TWCR & (1 << TWINT)));//wait for start to be transmitted

  if ((TWSR & 0xF8) != TW\_START)

    error\_led();

}

void twiWriteByte(uint8\_t DATA, uint8\_t type){

  TWDR = DATA;

  TWCR = \_BV(TWINT) | \_BV(TWEN);

  while (!(TWCR & (1 << TWINT))) {}

  if ((TWSR & 0xF8) != type)

    error\_led();

}

void twiAddr(uint8\_t addr, uint8\_t typeTWI){

  TWDR = addr;//send address

  TWCR = \_BV(TWINT) | \_BV(TWEN);    /\* clear interrupt to start transmission \*/

  while ((TWCR & \_BV(TWINT)) == 0); /\* wait for transmission \*/

  if ((TWSR & 0xF8) != typeTWI)

    error\_led();

}

voidwriteReg(uint8\_t reg, uint8\_t dat){

  //send start condition

  twiStart();

  twiAddr(camAddr\_WR, TW\_MT\_SLA\_ACK);

  twiWriteByte(reg, TW\_MT\_DATA\_ACK);

  twiWriteByte(dat, TW\_MT\_DATA\_ACK);

  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop

  \_delay\_ms(1);

}

static uint8\_t twiRd(uint8\_t nack){

  if (nack){

    TWCR = \_BV(TWINT) | \_BV(TWEN);

    while ((TWCR & \_BV(TWINT)) == 0); /\* wait for transmission \*/

    if ((TWSR & 0xF8) != TW\_MR\_DATA\_NACK)

      error\_led();

    return TWDR;

  }

  else{

    TWCR = \_BV(TWINT) | \_BV(TWEN) | \_BV(TWEA);

    while ((TWCR & \_BV(TWINT)) == 0); /\* wait for transmission \*/

    if ((TWSR & 0xF8) != TW\_MR\_DATA\_ACK)

      error\_led();

    return TWDR;

  }

}

uint8\_t rdReg(uint8\_t reg){

  uint8\_t dat;

  twiStart();

  twiAddr(camAddr\_WR, TW\_MT\_SLA\_ACK);

  twiWriteByte(reg, TW\_MT\_DATA\_ACK);

  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop

  \_delay\_ms(1);

  twiStart();

  twiAddr(camAddr\_RD, TW\_MR\_SLA\_ACK);

  dat = twiRd(1);

  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop

  \_delay\_ms(1);

  return dat;

}

void wrSensorRegs8\_8(const struct regval\_list reglist[]){

  uint8\_t reg\_addr, reg\_val;

  const struct regval\_list \*next = reglist;

  while ((reg\_addr != 0xff) | (reg\_val != 0xff)){

    reg\_addr = pgm\_read\_byte(&next->reg\_num);

    reg\_val = pgm\_read\_byte(&next->value);

   writeReg(reg\_addr, reg\_val);

    next++;

  }

}

void setColor(void){

  wrSensorRegs8\_8(yuv422\_ov7670);

 // wrSensorRegs8\_8(qvga\_ov7670);

}

void setResolution(void){

 writeReg(REG\_COM3, 4); // REG\_COM3 enable scaling

  wrSensorRegs8\_8(qvga\_ov7670);

}

void camInit(void){

 writeReg(0x12, 0x80);

  \_delay\_ms(100);

  wrSensorRegs8\_8(ov7670\_default\_regs);

 writeReg(REG\_COM10, 32);//PCLK does not toggle on HBLANK.

}

void arduinoUnoInut(void) {

  cli();//disable interrupts

    /\* Setup the 8mhz PWM clock

  \* This will be on pin 11\*/

  DDRB |= (1 << 3);//pin 11

  ASSR &= ~(\_BV(EXCLK) | \_BV(AS2));

  TCCR2A = (1 << COM2A0) | (1 << WGM21) | (1 << WGM20);

  TCCR2B = (1 << WGM22) | (1 << CS20);

  OCR2A = 0;//(F\_CPU)/(2\*(X+1))

  DDRC &= ~15;//low d0-d3 camera

  DDRD &= ~252;//d7-d4 and interrupt pins

  \_delay\_ms(3000);

    //set up twi for 100khz

  TWSR &= ~3;//disable prescaler for TWI

  TWBR = 72;//set to 100khz

    //enable serial

  UBRR0H = 0;

  UBRR0L = 1;//0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600 baud rate.

  UCSR0A |= 2;//double speed aysnc

  UCSR0B = (1 << RXEN0) | (1 << TXEN0);//Enable receiver and transmitter

  UCSR0C = 6;//async 1 stop bit 8bit char no parity bits

}

void StringPgm(const char \* str){

  do{

      while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit

      UDR0 = pgm\_read\_byte\_near(str);

      while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit

  } while (pgm\_read\_byte\_near(++str));

}

static void captureImg(uint16\_t wg, uint16\_t hg){

  uint16\_t y, x;

  StringPgm(PSTR("\*RDY\*"));

  while (!(PIND & 8));//wait for high

  while ((PIND & 8));//wait for low

    y = hg;

  while (y--){

        x = wg;

      //while (!(PIND & 256));//wait for high

    while (x--){

      while ((PIND & 4));//wait for low

            UDR0 = (PINC & 15) | (PIND & 240);

          while (!(UCSR0A & (1 << UDRE0)));//wait for byte to transmit

      while (!(PIND & 4));//wait for high

      while ((PIND & 4));//wait for low

      while (!(PIND & 4));//wait for high

    }

    //  while ((PIND & 256));//wait for low

  }

    \_delay\_ms(100);

}

void setup(){

  arduinoUnoInut();

  camInit();

  setResolution();

  setColor();

 writeReg(0x11, 10); //Earlier it had the value:writeReg(0x11, 12); New version works better for me :) !!!!

}

void loop(){

  captureImg(320, 240);

}

//https://randomnerdtutorials.com/arduino-lm35-lm335-lm34-temperature-sensor/

/\*

\* Rui Santos

\* Complete Project Details https://RandomNerdTutorials.com

\*/

const int sensorPin = A0;

float sensorValue;

float voltageOut;

float temperatureC;

float temperatureF;

// uncomment if using LM335

//float temperatureK;

void setup() {

pinMode(sensorPin, INPUT);

Serial.begin(9600);

}

void loop() {

sensorValue = analogRead(sensorPin);

voltageOut = (sensorValue \* 5000) / 1024;

// calculate temperature for LM35 (LM35DZ)

temperatureC = voltageOut / 10;

temperatureF = (temperatureC \* 1.8) + 32;

// calculate temperature for LM335

//temperatureK = voltageOut / 10;

//temperatureC = temperatureK - 273;

//temperatureF = (temperatureC \* 1.8) + 32;

// calculate temperature for LM34

//temperatureF = voltageOut / 10;

//temperatureC = (temperatureF - 32.0)\*(5.0/9.0);

Serial.print("Temperature(ºC): ");

Serial.print(temperatureC);

Serial.print(" Temperature(ºF): ");

Serial.print(temperatureF);

Serial.print(" Voltage(mV): ");

Serial.println(voltageOut);

delay(1000);

}