APPENDIX I

Marta Santamaria Santoyo

OV7670/OV7171 CMOS VGA (640x480) CAMERACHIP™ with OmniPixel® Technology

General Description

The OV7670/OV7171 CAMERACHIP™ is a low voltage CMOS image sensor that provides the full functionality of a single-chip VGA camera and image processor in a small footprint package. The OV7670/OV7171 provides full-frame, sub-sampled or windowed 8-bit images in a wide range of formats, controlled through the Serial Camera Control Bus (SCCB) interface.

This product has an image array capable of operating at up to 30 frames per second (fps) in VGA with complete user control over image quality, formatting and output data transfer. All required image processing functions, including exposure control, gamma, white balance, color saturation, hue control and more, are also programmable through the SCCB interface. In addition, OmniVision CAMERACHIPS use proprietary sensor technology to improve image quality by reducing or eliminating common lighting/electrical sources of image contamination, such as fixed pattern noise (FPN), smearing, blooming, etc., to produce a clean, fully stable color image.

Note: The OV7670/OV7171 uses a lead-free package.

Features

- High sensitivity for low-light operation
- Low operating voltage for embedded portable apps
- Standard SCCB interface compatible with I2C interface
- Output support for Raw RGB, RGB (GRB 4:2:2, RGB565/555/444), YUV (4:2:2) and YCbCr (4:2:2) formats
- Supports image sizes: VGA, CIF, and any size scaling down from CIF to 40x30
- VarioPixel® method for sub-sampling
- Automatic image control functions including: Automatic Exposure Control (AEC), Automatic Gain Control (AGC), Automatic White Balance (AWB), Automatic Band Filter (ABF), and Automatic Black-Level Calibration (ABLC)
- Image quality controls including color saturation, hue, gamma, sharpness (edge enhancement), and anti-blooming
- ISP includes noise reduction and defect correction
- Supports LED and flash strobe mode
- Supports scaling
- Lens shading correction
- Flicker (50/60 Hz) auto detection
- Saturation level auto adjust (UV adjust)
- Edge enhancement level auto adjust
- De-noise level auto adjust

Ordering Information

Product	Package
OV07670-VL2A (Color, lead-free)	24 pin CSP2
OV07171-VL2A (B&W, lead-free)	24 pin CSP2

Applications

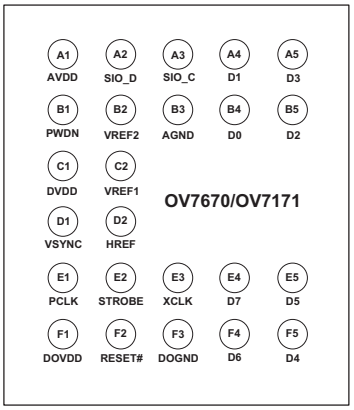
- Cellular and Picture Phones
- Toys
- PC Multimedia
- Digital Still Cameras

Key Specifications

Active Array Size		640 x 480
Power Supply	**Digital Core**	1.8VDC ±10%
	Analog	2.45V to 3.0V
	I/O	1.7V to 3.0V[a]
Power Requirements	**Active**	60 mW typical (15fps VGA YUV format)
	Standby	< 20 µA
Temperature Range	**Operation**	-30°C to 70°C
	Stable Image	0°C to 50°C
Output Formats (8-bit)		• YUV/YCbCr 4:2:2 • RGB565/555/444 • GRB 4:2:2 • Raw RGB Data
Lens Size		1/6"
Chief Ray Angle		25°
Maximum Image Transfer Rate		30 fps for VGA
Sensitivity		1.3 V/(Lux • sec)
S/N Ratio		46 dB
Dynamic Range		52 dB
Scan Mode		Progressive
Electronics Exposure		Up to 510:1 (for selected fps)
Pixel Size		3.6 µm x 3.6 µm
Dark Current		12 mV/s at 60°C
Well Capacity		17 K e
Image Area		2.36 mm x 1.76 mm
Package Dimensions		3785 µm x 4235 µm

a. I/O power should be 2.45V or higher when using the internal regulator for Core (1.8V); otherwise, it is necessary to provide an external 1.8V for the Core power supply.

Figure 1 OV7670/OV7171 Pin Diagram (Top View)



Functional Description

Figure 2 shows the functional block diagram of the OV7670/OV7171 image sensor. The OV7670/OV7171 includes:

- Image Sensor Array (total array of 656 x 488 pixels, with active pixels 640 x 480 in YUV mode)
- Analog Signal Processor
- A/D Converters
- Test Pattern Generator
- Digital Signal Processor (DSP)
- Image Scaler
- Timing Generator
- Digital Video Port
- SCCB Interface
- LED and Strobe Flash Control Output

Figure 2 Functional Block Diagram

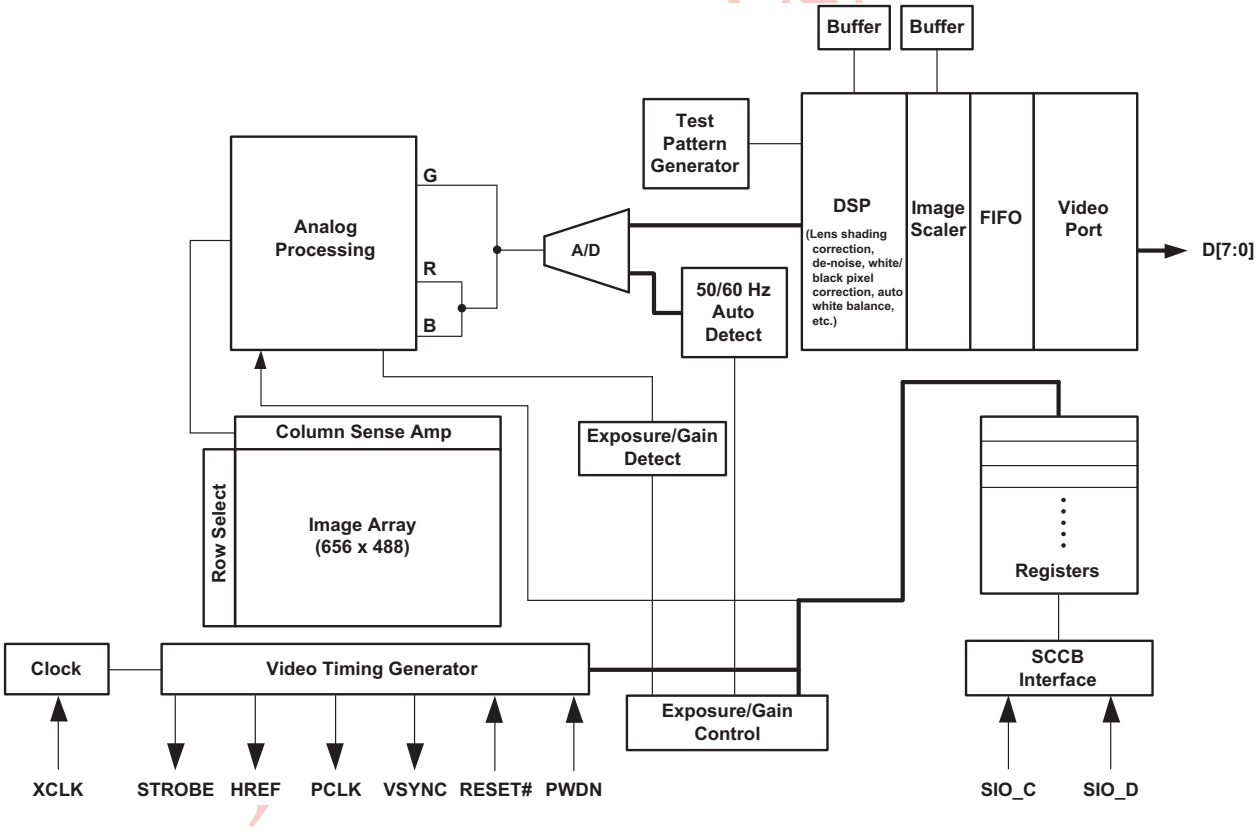
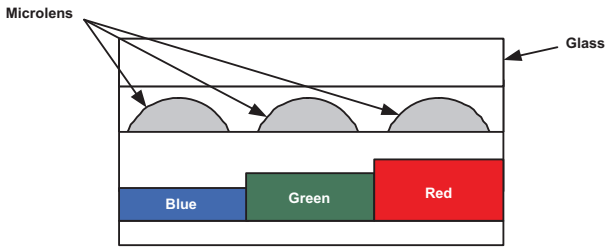


Image Sensor Array

The OV7670/OV7171 sensor has an image array of 656 x 488 pixels for a total of 320,128 pixels, of which 640 x 480 pixels are active (307,200 pixels). Figure 3 shows a cross-section of the image sensor array.

Figure 3 Image Sensor Array



Timing Generator

In general, the timing generator controls the following functions:

- Array control and frame generation
- Internal timing signal generation and distribution
- Frame rate timing
- Automatic Exposure Control (AEC)
- External timing outputs (VSYNC, HREF/HSYNC, and PCLK)

Analog Signal Processor

This block performs all analog image functions including:

- Automatic Gain Control (AGC)
- Automatic White Balance (AWB)

A/D Converters

After the Analog Processing block, the bayer pattern Raw signal is fed to a 10-bit analog-to-digital (A/D) converter shared by G and BR channels. This A/D converter operates at speeds up to 12 MHz and is fully synchronous to the pixel rate (actual conversion rate is related to the frame rate).

In addition to the A/D conversion, this block also has the following functions:

- Digital Black-Level Calibration (BLC)
- Optional U/V channel delay
- Additional A/D range controls

In general, the combination of the A/D Range Multiplier and A/D Range Control sets the A/D range and maximum value to allow the user to adjust the final image brightness as a function of the individual application.

Test Pattern Generator

The Test Pattern Generator features the following:

- 8-bar color bar pattern
- Fade-to-gray color bar pattern
- Shift "1" in output pin

Digital Signal Processor (DSP)

This block controls the interpolation from Raw data to RGB and some image quality control.

- Edge enhancement (a two-dimensional high pass filter)
- Color space converter (can change Raw data to RGB or YUV/YCbCr)
- RGB matrix to eliminate color cross talk
- Hue and saturation control
- White/black pixel correction
- De-noise
- Lens shading correction
- Programmable gamma control
- Transfer 10-bit data to 8-bit

Image Scaler

This block controls all output and data formatting required prior to sending the image out. This block scales YUV/RGB output from VGA to CIF and almost any size under CIF.

Digital Video Port

Register bits COM2[1:0] increase I_{OL}/I_{OH} drive current and can be adjusted as a function of the customer's loading.

SCCB Interface

The Serial Camera Control Bus (SCCB) interface controls the CAMERACHIP operation. Refer to *OmniVision Technologies Serial Camera Control Bus (SCCB) Specification* for detailed usage of the serial control port.

LED and Strobe Flash Control Output

The OV7670/OV7171 has a Strobe mode that allows it to work with an external flash and LED.

Pin Description

Table 1 Pin Description

Pin Number	Name	Pin Type	Function/Description
A1	AVDD	Power	Analog power supply
A2	SIO_D	I/O	SCCB serial interface data I/O
A3	SIO_C	Input	SCCB serial interface clock input
A4	D1[a]	Output	YUV/RGB video component output bit[1]
A5	D3	Output	YUV/RGB video component output bit[3]
B1	PWDN	Input (0)[b]	Power Down Mode Selection 0: Normal mode 1: Power down mode
B2	VREF2	Reference	Reference voltage - connect to ground using a 0.1 µF capacitor
B3	AGND	Power	Analog ground
B4	D0	Output	YUV/RGB video component output bit[0]
B5	D2	Output	YUV/RGB video component output bit[2]
C1	DVDD	Power	Power supply (+1.8 VDC) for digital logic core
C2	VREF1	Reference	Reference voltage - connect to ground using a 0.1 µF capacitor
D1	VSYNC	Output	Vertical sync output
D2	HREF	Output	HREF output
E1	PCLK	Output	Pixel clock output
E2	STROBE	Output	LED/strobe control output
E3	XCLK	Input	System clock input
E4	D7	Output	YUV/RGB video component output bit[7]
E5	D5	Output	YUV/RGB video component output bit[5]
F1	DOVDD	Power	Digital power supply for I/O (1.7V ~ 3.0V)
F2	RESET#	Input	Clears all registers and resets them to their default values. 0: Reset mode 1: Normal mode
F3	DOGND	Power	Digital ground
F4	D6	Output	YUV/RGB video component output bit[6]
F5	D4	Output	YUV/RGB video component output bit[4]

a. D[7:0] for 8-bit YUV or RGB (D[7] MSB, D[0] LSB)

b. Input (0) represents an internal pull-down resistor.

APPENDIX II

Marta Santamaria Santoyo

APPENDIX II_ Software for the QR Codes Decoder

Main Software for the QR Codes Decoder (CCS C Compiler, C Language)

```
/*********************************************************************/

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

/********** MAIN SOFTWARE FOR THE QR CODES DECODER  **********/

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

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

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

#include <18F4550.h>

#fuses
HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN

#use delay(clock=48000000)

#use i2c (master, sda=PIN_B0, scl=PIN_B1, FAST=400000)

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

#DEFINE USB_HID_DEVICE TRUE                    /* ACTIVATES THE HID
                                                  PROTOCOL*/

#define USB_EP1_TX_ENABLE USB_ENABLE_INTERRUPT /* DEFINITION OF
                                                  THE OUTPUT  BUFFER SIZE*/
#define USB_EP1_TX_SIZE 8

#define USB_EP1_RX_ENABLE USB_ENABLE_INTERRUPT /* DEFINITION OF
                                                  THE INPUT BUFFER SIZE*/
#define USB_EP1_RX_SIZE 8

#include <pic18_usb.h>      /*LOW LEVEL FUNCTIONS (HARDWARE) FOR THE
                                                  PIC 18FXX5X SERIES THAT WILL BE
                                                  HELPFUL ON THE usb.c*/
#include <./usb_kbd_HID.h>  /*DESCRIPTIONS OF THE DEVICE (HOW
                                                  WINDOWS WILL RECOGNIZE IT)*/
#include <usb.c>           //LIBRARY FOR THE USE OF USB


#define FRRST  PIN_A0

#define FOE    PIN_A1
```

```
#define FWEN   PIN_A2
#define FCLK   PIN_A3
#define END_FRAME    PIN_B5
#define RESET_FIFO   PIN_B6
#define TAKE_PICTURE PIN_B7
#define SIO_C  PIN_B1
#define SIO_D  PIN_B0
#define OV7670_ID 0X7F
#define COM3   0X0C
#define COM7   0X12

int out_data[20];
int in_data[2];
int send_timer;
int contador;
int counter;
int counter_h;
int counter_v;
int pixel;

//INITIALIZATION OF OV7670
void initialize_OV7670 (){
  // Configuration for QCIF format
     // COM3 register: Enable format scaling
     i2c_start();
     i2c_write(OV7670_ID);
     i2c_write(COM3);
```

```
      i2c_write(0b00001000);
       delay_us(2);
       // COM7 register: Select QCIF format and YUV
      i2c_write(OV7670_ID);
      i2c_write(COM7);
      i2c_write(0b00001000);
      delay_us(2);
      i2c_stop();
}

//Initialize FIFO
 void initialize_fifo(){
  OUTPUT_LOW (FOE);
  OUTPUT_LOW(FRRST);
  delay_us(2);
  OUTPUT_HIGH(FRRST);
}

// Interrupts Configuration
#int_rb
void kbis_isr(){
 // TAKE PICTURE
   if (input(TAKE_PICTURE)){
      initialize_fifo();
      OUTPUT_LOW(FWEN);
   }
      // END FRAME
```

```
  if (input(END_FRAME)){
    // SENDS TO THE PROGRAM TO START RECORDING
   out_data[1] = 0XDE;
     usb_puts(1,out_data,2,100);
     out_data[1] = 0XAD;
     usb_puts(1,out_data,2,100);
     out_data[1] = 0XFA;
     usb_puts(1,out_data,2,100);
     out_data[1] = 0XCE;
     usb_puts(1,out_data,2,100);
     OUTPUT_HIGH(FWEN);
    OUTPUT_LOW (FRRST);
    delay_us(2);
    OUTPUT_HIGH(FRRST);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
  }

  //RESET FIFO
  if (input(RESET_FIFO)){
  initialize_fifo();
  }
}

//Timer configuration
#int_timer0
void int_timer(){
   counter++;
```

```
   if(counter == 2) {
       counter=0;
       if (counter_h <= 176){
         counter_v++;
         output_low(FOE);
         delay_us(10);
         output_low(FRRST);
         delay_us(2);
         output_high(FRRST);
         delay_us(2);
         pixel=input_d();
         out_data[1] = pixel;
         usb_puts(1,out_data,2,100);
         if (counter_v >=144){
         counter_v=0;
         counter_h++;
         }
       }       else{
       OUTPUT_HIGH (FOE);
       setup_timer_0(RTCC_OFF);
          /* SENDS IT TO THE PROGRAM TO SAY IT HAS FINISHED
RECORDING*/
      out_data[1] = 0XFA;
     usb_puts(1,out_data,2,100);
     out_data[1] = 0XCE;
     usb_puts(1,out_data,2,100);
     out_data[1] = 0X0F;
```

```
      usb_puts(1,out_data,2,100);
      out_data[1] = 0X0F;
      usb_puts(1,out_data,2,100);
        }
      }
}
//MAIN LOOP
void main(){
  // INITIALIZATIONS
  send_timer=0;
  contador=0;
  counter=0;
  counter_h=0;
  counter_v=0;
 // PORTS CONFIGURATION
  set_tris_d(0x00);
  set_tris_b(0xff);
  delay_ms(500);
   // INTERRUPT CONFIGURATION
  clear_interrupt(int_rb); // Clear interrupt flag
  enable_interrupts(global); // Enable global interrupts
  enable_interrupts(int_rb); // Enable Interrupts on change
  ext_int_edge(H_TO_L); // Interruption Flag High to Low
  initialize_OV7670 (); //Initialize OV7670
 //USB CONFIGURATION
  usb_init(); //INITALIZATES AND WAITS TILL IT IS RECOGNIZED
  usb_task();
```

```
  usb_wait_for_enumeration(); //WAITS TILL IT IS ENUMERATED (RECOGNIZED BY THE PC)

    while(TRUE){

     // INFINITE LOOP

       }

}


/*********************************END ********************************/
```

Software for the QR Codes Decoder (Visual Basics, Basic Language)

```
Option Explicit On

Imports System.IO
Imports System
Imports System.Text


Public Class frmUSB
    ' contadores y variables
    Public contador1 As Integer
    Public contador2 As Integer
    Public contador3 As Long
    Public grabar As Boolean
    Public wfile As System.IO.FileStream
    Public bytedata(1) As Byte




    ' vendor and product IDs
    Private Const VendorID As Integer = &H4D8    'Replace with your device's
    Private Const ProductID As Integer = &H0      'product and vendor IDs

    ' read and write buffers
    Private Const BufferInSize As Integer = 1 'Size of the data buffer coming IN to the PC
    Private Const BufferOutSize As Integer = 1    'Size of the data buffer going OUT from the PC
    Dim BufferIn(BufferInSize) As Byte          'Received data will be stored here - the first byte in the array is unused
```

```
    Dim BufferOut(BufferOutSize) As Byte    'Transmitted data is stored here - the first
item in the array must be 0

    ' ***************************************************************
    ' when the form loads, connect to the HID controller - pass
    ' the form window handle so that you can receive notification
    ' events...
    '***************************************************************
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' do not remove!
        ConnectToHID(Me)
        grabar = False
        contador1 = 0
        contador2 = 0
        contador3 = 0
    End Sub

    '***************************************************************
    ' disconnect from the HID controller...
    '***************************************************************
    Private Sub Form1_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        DisconnectFromHID()
    End Sub

    '***************************************************************
    ' a HID device has been plugged in...
    '***************************************************************
    Public Sub OnPlugged(ByVal pHandle As Integer)
        If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
            ' ** YOUR CODE HERE **
            Label1.Text = "Connected"

        End If
    End Sub

    '***************************************************************
    ' a HID device has been unplugged...
    '***************************************************************
    Public Sub OnUnplugged(ByVal pHandle As Integer)
        If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle) =
ProductID Then
            hidSetReadNotify(hidGetHandle(VendorID, ProductID), False)
            ' ** YOUR CODE HERE **
            Label1.Text = "Not connected"
        End If
    End Sub
```

```
'*********************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*********************************************************************
Public Sub OnChanged()
   ' get the handle of the device we are interested in, then set
   ' its read notify flag to true - this ensures you get a read
   ' notification message when there is some data to read...
   Dim pHandle As Integer
   pHandle = hidGetHandle(VendorID, ProductID)
   hidSetReadNotify(hidGetHandle(VendorID, ProductID), True)
End Sub

'*********************************************************************
' on read event...
'*********************************************************************
Public Sub OnRead(ByVal pHandle As Integer)
   ' read the data (don't forget, pass the whole array)...
   If hidRead(pHandle, BufferIn(0)) Then
      ' ** YOUR CODE HERE **
      ' first byte is the report ID, e.g. BufferIn(0)
      ' the other bytes are the data from the microcontroller...

      ' Recibimos datos
      TextBox2.Text = TextBox2.Text & BufferIn(1)

      ' Estamos grabando?
      If (grabar) Then
         ' aqui escribimos a archivo
         TextBox1.Text = "Recording file..."
         bytedata(0) = BufferIn(1)

         wfile.Write(bytedata, contador3, 1)
         contador3 = contador3 + 1
         If (contador2 = 0 And BufferIn(1) = &HFA) Then
            contador2 = contador2 + 1
         End If
         If (contador2 = 1 And BufferIn(1) = &HCE) Then
            contador2 = contador2 + 1
         End If
         If (contador2 = 2 And BufferIn(1) = &HF) Then
            contador2 = contador2 + 1
         End If
         If (contador2 = 3 And BufferIn(1) = &HF) Then
            contador2 = contador2 + 1
         End If

         If (contador2 = 4) Then
            contador2 = 0
            'stop recording, warning, the 4 last bytes of the file must be
```

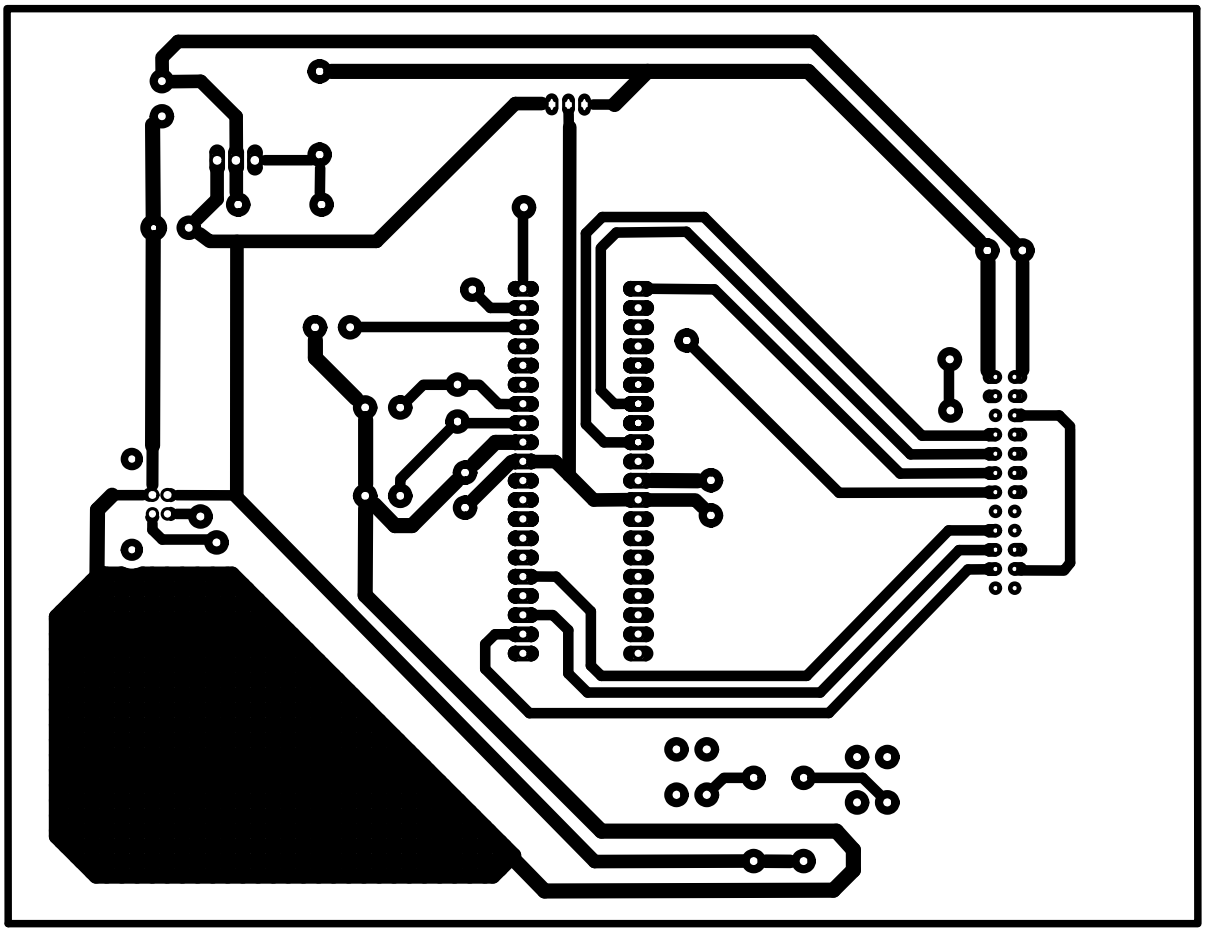
```
                'deleted, they are the finalization string
                grabar = False
                ' close the file
                wfile.Close()
            End If

        Else
            ' waiting
            TextBox1.Text = "Waiting..."

            ' the counter is increased if the initialization string is being received

            If (contador1 = 0 And BufferIn(1) = &HDE) Then
                contador1 = contador1 + 1
            End If
            If (contador1 = 1 And BufferIn(1) = &HAD) Then
                contador1 = contador1 + 1
            End If
            If (contador1 = 2 And BufferIn(1) = &HFA) Then
                contador1 = contador1 + 1
            End If
            If (contador1 = 3 And BufferIn(1) = &HCE) Then
                contador1 = contador1 + 1
            End If
        End If

        If (contador1 = 4) Then
            grabar = True
            ' the file is open for recording
            wfile = New FileStream("imagen.img", FileMode.Open, FileAccess.Write)


        End If


    End If
  End Sub

  Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs)

  End Sub

  Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label1.Click

  End Sub
End Class

' END
```

APPENDIX III

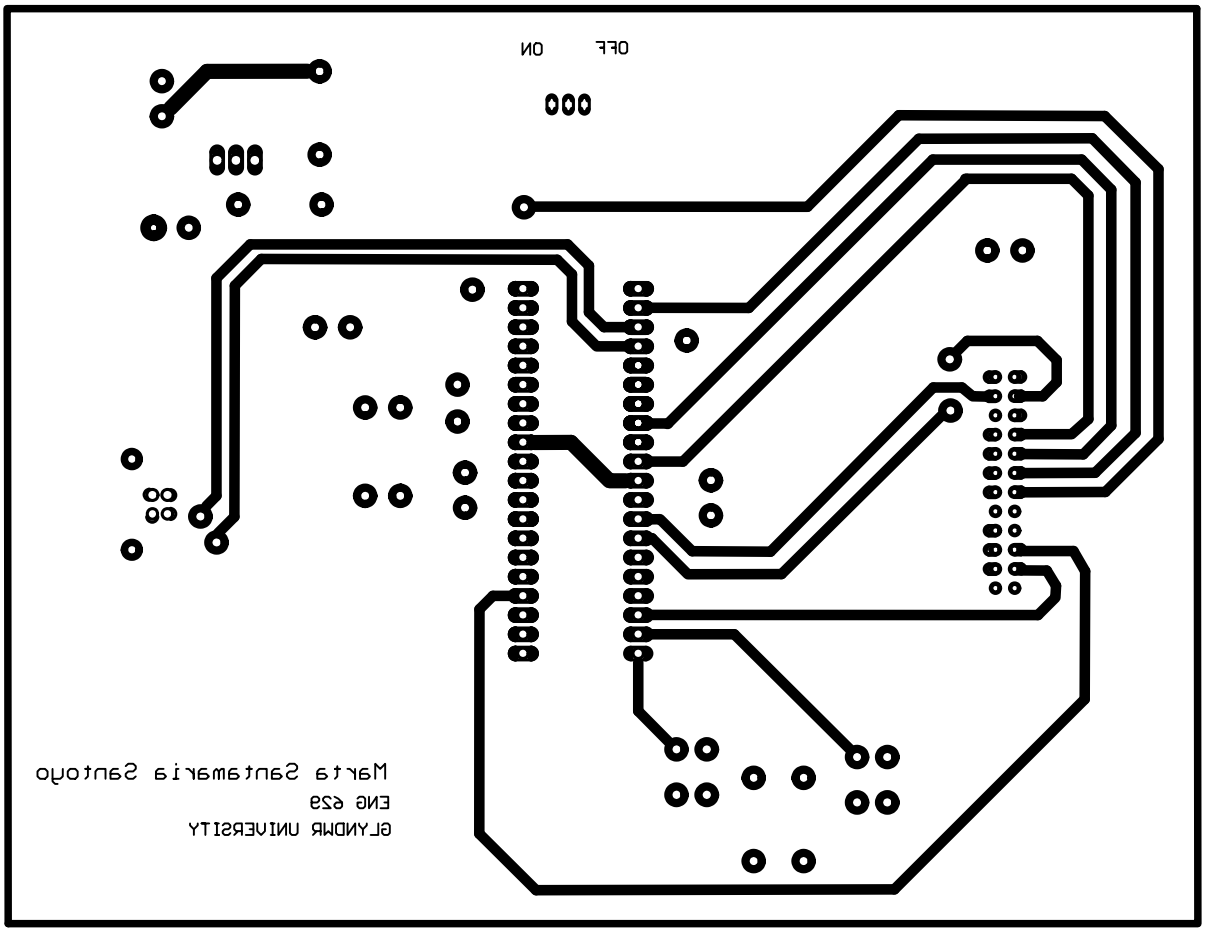
Marta Santamaria Santoyo

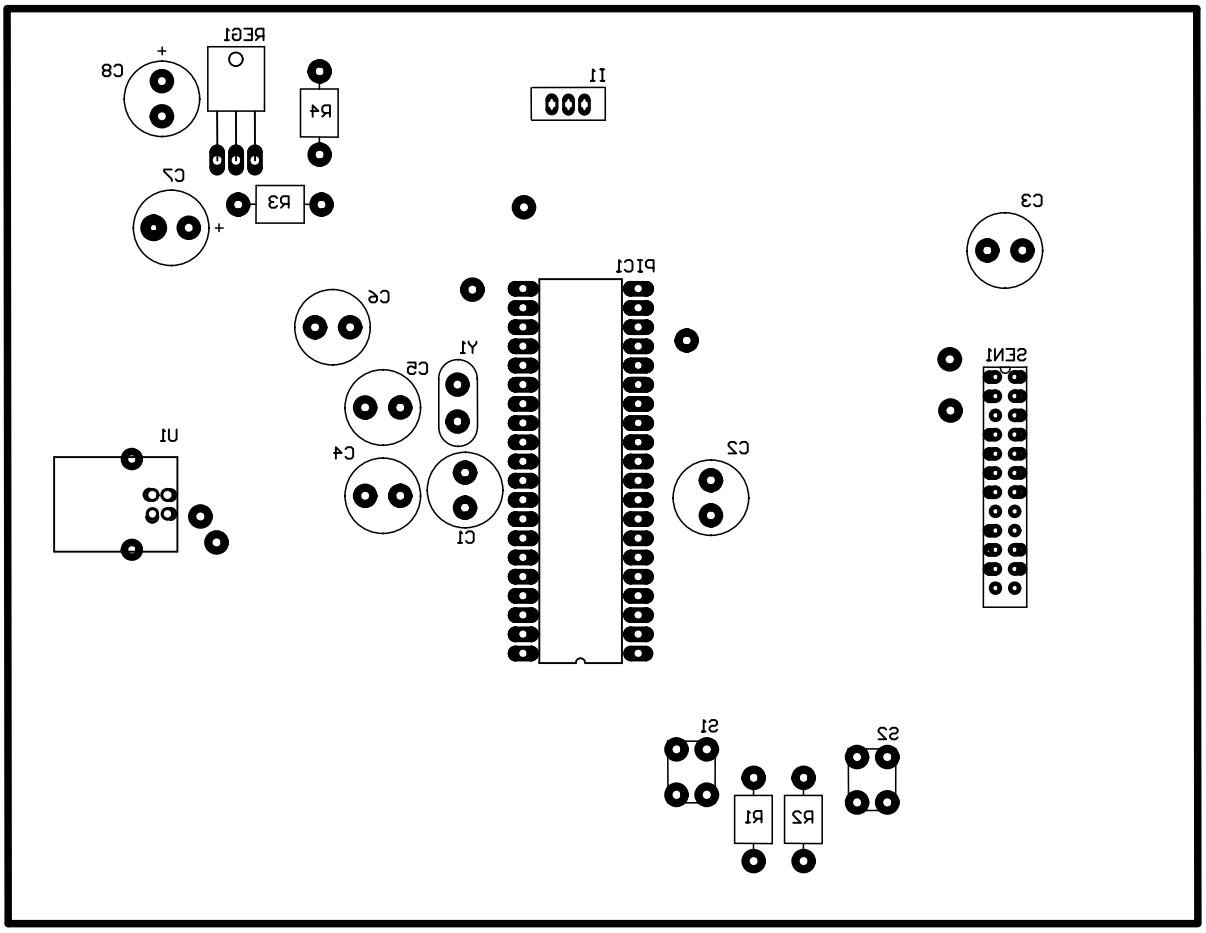
APPENDIX III_ Final Artworks of the PCB (1:1 Scaling)

Bottom Layer



Top Layer


ON
OFF
Marta Santamaría Santoyo
ENG 629
GLYNDWR UNIVERSITY


REG1
C8
R4
J1
C7
R3
C3
PIC1
C6
Y1
C5
SEN1
U1
C4
C2
C1
S1
S2
R1
R2