DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINERING

EEE-491
Electrical and Electronics Engineering Design I

Spoken Number Recognition Project

FINAL LAB REPORT

MUSTEBA ANIL ONDER

HAZRAT POLADOV

OGUZHAN TOR

31.05.2022

TABLE OF CONTENTS

1 PROJECT OBJECTIVE AND SPECIFICATIONS

The goal of the project is to create a system that can detect and display numbers from 0 to 9 when they are spoken to it via sound signals by a human (analog signals). The system must be created utilizing an FPGA board (Basys-3) and the Hardware Description Language (HDL) (VHDL). The project's goal is to achieve maximum accuracy in identifying and presenting numbers based on the information provided to the system. According to the labs, the following is a list of system specifications:

1.1 Lab - DEBUG

A debugger that can send 32-bit data to a MATLAB application running on a PC via USB connection must be devised and implemented. Because the Universal Asynchronous Receiver and Transmitter (UART) can only transmit and receive 8-bit data in a transaction, a Finite State Machine (FSM) must be constructed to trigger the UART four times in order to transfer and receive 32-bit data.

1.2 Lab - ADC

It is necessary to develop and build a Serial Peripheral Interface (SPI) to an analog to digital converter (ADC) board. To sample analog signals at the ADC's input, the subsystem will use a 12 bit ADC board module. The SPI interface must run at a clock rate of 12.5 Mhz, with a sampling rate of 500K samples per second and decimation reducing the sampling rate by a factor of 32.

1.3 Lab - WINDOW

From the Lab ADC module, the window module will read 512 samples of 12 bit data. The RAM's starting address will be 14 bits, with 512 samples of data (a frame) multiplied by 512 coefficients of the HANNING windowing function as a starting parameter. Each coefficient data point must be 8 bits, and the multiplication result must be 20 bits.

1.4 Lab - PCB

It is necessary to design and print a microphone amplifier, an anti-aliasing filter circuit, and a printed circuit board (PCB). A single PCB with a maximum area of 25 cm2 must be used for the design. The anti-aliasing filter will feature cascaded Sallen - Key Low - Pass Filter (LPF) stages with an attenuation of 80 dB per decade and a gain of -1 to +1 dB from 100 Hz to 3 kHz. In addition, the LPF's 3 dB corner frequency must be between 4 and 5 kHz.

1.5 Lab - CTRL

A data flow control block, as well as six building blocks (subsystems) with control signals, "start" as an input and "ready" as an output, must be created and implemented. The control block will also divide the voice signal (which is stored in the Lab - ADC dual-port RAM) into frames by sending the frame start address to the Lab - WINDOW. Each frame control block must provide a "start" signal to each subsystem in which the frames must be 50 percent overlapped.

1.6 Lab - FFT

The Fast Fourier Transform (FFT) block will read 512 samples of 20-bit data from the dual-port RAM of the Window block and then perform a 512-point FFT algorithm. The squared magnitude from an FFT output with a real component of 30 bits and an imaginary part of 30 bits must be computed and then trimmed to 32 bits before being written into a basic dual-port RAM.

1.7 Lab - MEL

The Mel Filter block will read 256-bit data from the dual-port RAM of the FFT block. Where the frequency ranges from 100 Hz to 4 kHz, at least 13 MEL filters must be used.

1.8 Lab - MATLAB

The total system will be built utilizing the Mel - Frequency Cepstrum in the MATLAB application. The sound signals should be captured using a PC microphone, and the data should be saved in MATLAB. Each spoken number must be divided into 30 millisecond audio frames that are roughly 50 percent overlapped (15 milliseconds).

1.9 Lab - DCT

The Discrete Cosine Transform (DCT) block will read energy data from MEL block's dual-port RAM for each frame and then compute DCT of the read data. The findings will be saved in a basic dual-port RAM with a 16-bit size.

1.10 Lab - COMPARE

The comparison block has two modes of operation: "record" and "compare." The feature vectors of a spoken number must be kept in a RAM called REF - RAM in the record mode. The spoken number will be recognized by a four-bit number input signal that specifies the start address of the spoken number's feature vectors in REF-RAM. The feature vectors must store all ten digits from 0 to 9. In the comparison mode, feature vectors of the spoken numbers are recorded in COMP - RAM, and subsequently Euler distances between COMP - RAM contents and REF - RAM contents are calculated for each number. The compare block will then choose the closest one to display on the FPGA board's seven-segment display.

2 SYSTEM DESCRIPTION

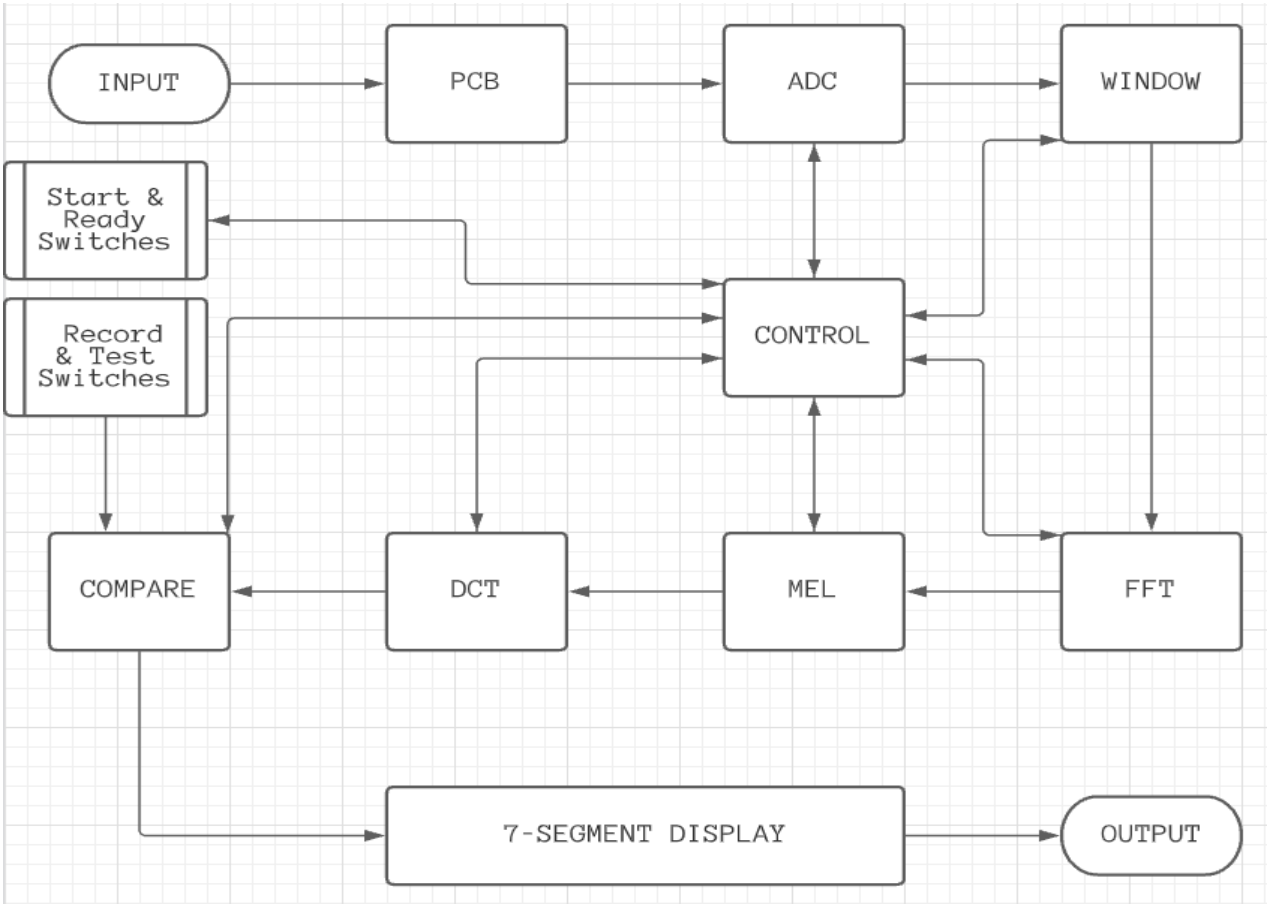


Figure 1: Block Diagram of the System

The PCB's microphone receives analog signals (sound signals), which are amplified and then filtered by Sallen - Key Low - Pass filter stages in a cascade. Signals are transmitted to the ADC after the PCB process. The ADC block uses SPI and sampling to convert analog signals to digital signals. The Window block reads the sampled 16384 data, which is then filtered by the Hanning Windows. After processing the data through 32 frames, the 12 bit input data is multiplied by the 8 bit Hanning Coefficients, and the 20 bit output data is generated. The FFT block reads 512 samples of 20-bit data, which is subsequently squared and truncated to 32-bit data. In the MEL block, when triangular-shaped waves are seen, the FFT data are filtered by 13 MEL filters.

The energy data of MEL filters are then computed in the DCT block, followed by Euler distances in the Compare block to determine which number is spoken over the microphone.

The "ready" and "start" signals of each block are used to regulate the data flow in the control block. All data transfers are read and written in each block's dual-port RAMs. The outcome of the Compare block's operation will be presented in the 7 - Segment Display at the conclusion.

3 IMPLEMENTATION AND TESTS

3.1 Debug Module

The Debug Module, the project's initial module, is responsible for transferring 32-bit data to Matlab on the computer. We will be able to utilize it in this way to ensure that each module built in subsequent phases of the project is functional and to discover mistakes.

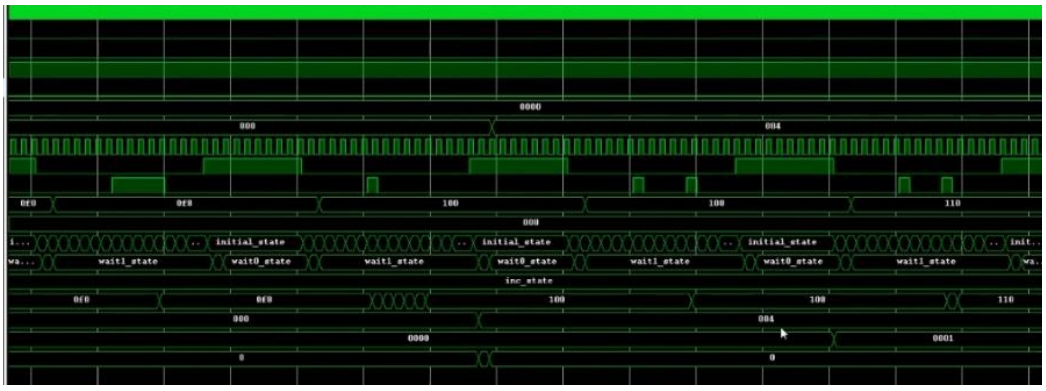
We utilized the Universal Asynchronous Receiver and Transmitter (UART) protocol within the debug module to transport data from FPGA to Matlab, and we used pre-written UART code [1]. However, UART can only transfer 8 bits at a time, when we need to send 32 bits. As a result, in the FSM algorithm, we'll deliver 32-bit data in four phases. First, the first MSB will be supplied with eight bits, followed by the second MSB with eight bits, and so on.

You have 14 addresses to choose from, as indicated in the Assignment. This indicates that 16384 32-bit data will be sent. There are two 32-bit headers, start and stop, in addition to these. AA5503CC and 55AACC03, respectively. We'll start transferring data when the start header arrives, and after the transfer is complete, we'll send the stop header.

3.2 ADC Module

The human voice, which is an analog signal, must be transformed into a digital signal before the FPGA can identify it. This is why we need the ADC module. The A Serial Peripheral Interface will be used to communicate between the ADC board and the FPGA (SPI). The ADC module we're using at this point will sample 12 bits of data and has a 12.5 Mhz SPI clock rate. This indicates that the ADC can sample at a rate of around 500K samples per second.

On our test bench, we established a data count and spi array to test the ADC module. The data count reflects the data that is transferred to SPI, and we store it in the spi array. We continue by raising the data count by 1 each time, and when the data count falls below 16, we transmit 111 in hexadecimal. We transmit 333 in hexadecimal when the number is between 16 and 32. We transmit 555 in hexadecimal when the value is between 32 and 48. We send 777 in all other circumstances.



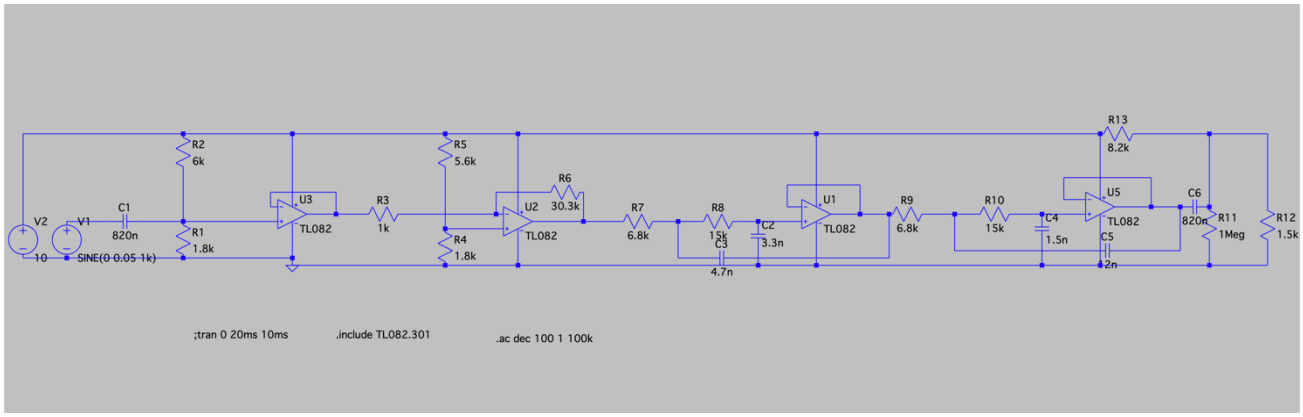
3.3 Window Module

We transmit the signal via the windows in order to reduce the leakage problem in the FFT module throughout the signal processing phases, therefore the Lab Window is quite crucial in this project. We attenuate the signal's beginning and ends while amplifying the center. As a consequence, in FFT, our signal will be sharper.

The 12-bit data from the ADC is passed through 512 Hanning windows to accomplish this. This indicates that samples are multiplied by 512 Hanning coefficients. The output of the multiplication is 20 bits of data since Hanning coefficients are 8 bits and data is 12 bits. This windowing method is applied to each of the 16384 addresses. The 20-bit data is then written to a basic dual-port RAM.

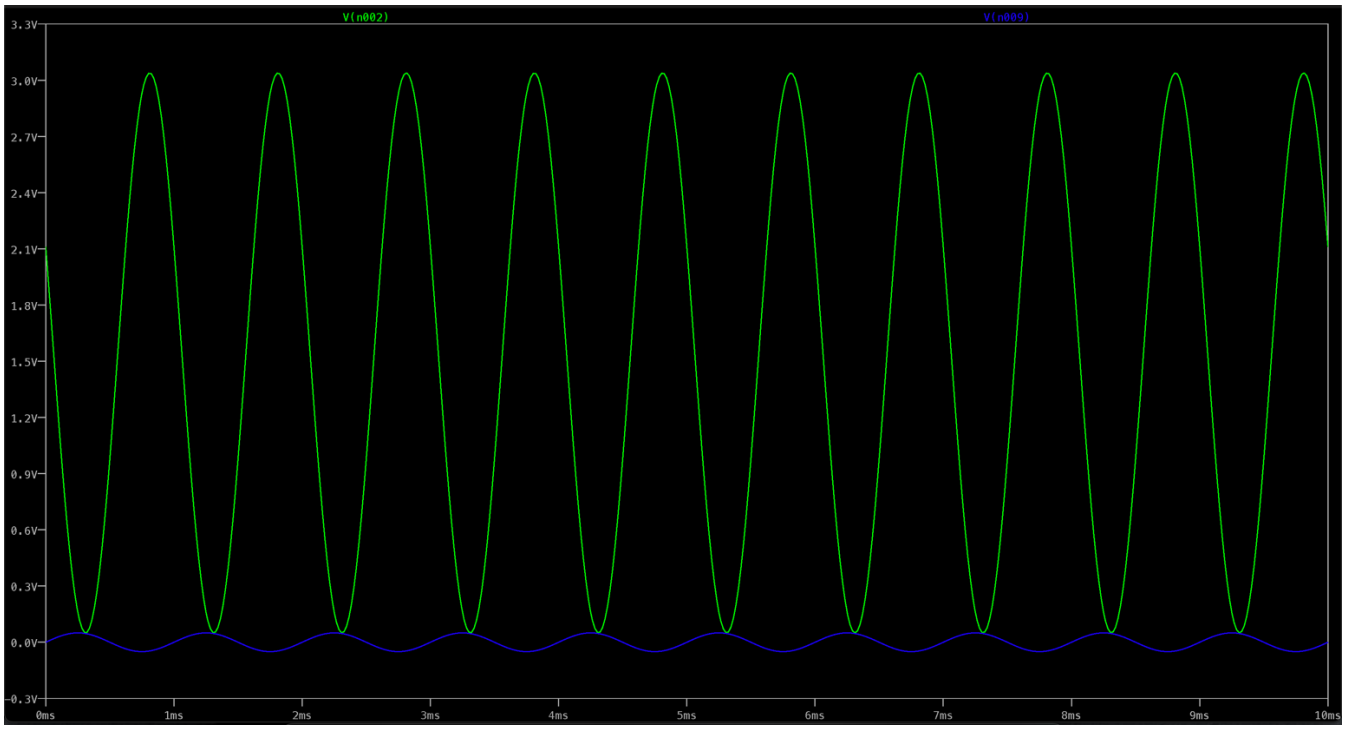
To test the Window module, we assumed the ADC data was constant as described in the assignment and wrote it to the test bench in this manner. We used Matlab's hanning function to produce 512 coefficients, which we quantified to 8 bits. Then, using a multiplier, we multiplied the ADC data by 512 coefficients and saved the result in RAM.

3.4 PCB Module



The first stage is a buffer that just adds a DC offset before sending the signal to the second stage. The second step is an amplifier, which boosts the signal by around 30 decibels (figure 2). The audio amplifier's needed bandwidth, on the other hand, is not met. As a result, two Sallen Key low pass filters are employed in this scenario. One of these is a second-degree LPF, while the other two combine to form a fourth-degree LPF. Finally, after coupling (separating DC from AC), we must modify the DC offset such that the amplified and filtered signal is maximum 3.3 V and minimum 0 V before sending it to the ADC. Nodal analysis and standard OPAMP criteria were used in all operations.

As a result, in PCB design, negative voltage is not a possibility; all Op Amps work in positive areas. The supply voltage is 10 V, and there are two voltage dividers in each of the first two stages, as well as one after the coupling.


V(n002)
V(n009)
3.3V
3.0V
2.7V
2.4V
2.1V
1.8V
1.5V
1.2V
0.9V
0.6V
0.3V
0.0V
-0.3V
0ms
1ms
2ms
3ms
4ms
5ms
6ms
7ms
8ms
9ms
10ms

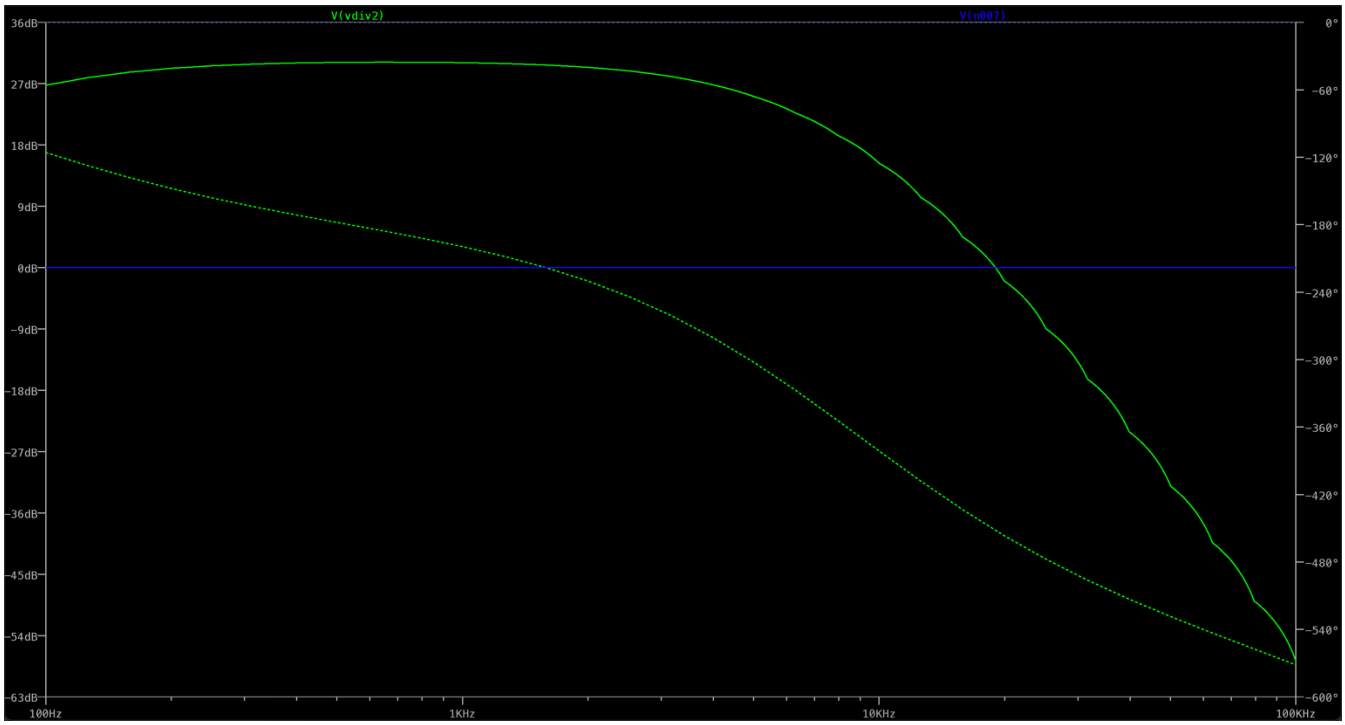
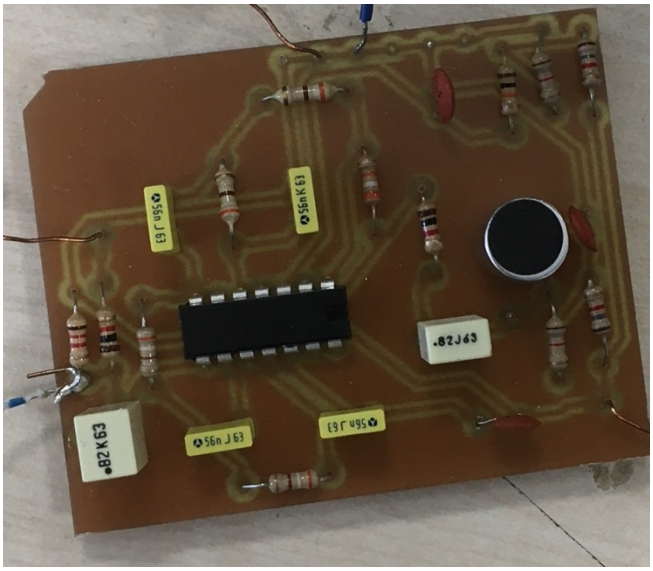

V(vdiv2)
V(n007)
36dB
27dB
18dB
9dB
0dB
-9dB
-18dB
-27dB
-36dB
-45dB
-54dB
-63dB
0°
-60°
-120°
-180°
-240°
-300°
-360°
-420°
-480°
-540°
-600°
100Hz
1KHz
10KHz
100KHz

Figure 4 is the AC analysis of output. The 3 dB drop happens at 3.95kHz and decade attenuation is 80 dB as can be seen from figure 4.



3.5 Matlab Module

Using the MATLAB application, we developed and programmed practically the entire system. We utilized the microphone on a computer to collect sound signals, which were then analyzed to determine which number was uttered to the system. We are storing audio signals as a frame of 30 milliseconds duration, as indicated in the lab handbook, and then, as we did in the Lab – CONTROL, these frames are overlapped around 50%. The FFTs of stored data are calculated once they are multiplied by Hanning Coefficients. After applying the 13 MEL filters to these frames, their DCTs are obtained to obtain feature vectors. For each number from 0 to 9, the obtained feature vectors are kept in a matrix. Following the storage of feature vectors, whenever a new number is given to the system, it is recorded in a new matrix, and the Euclidean distance between the pre-stored vector features is used to determine which number was given to the system. To make the system compatible with the ADC block output, we saved our data as 12 bits, as required in our real hardware system.

BURAYA MATLAB FOTOLARI GELCEK

3.6 Control Module

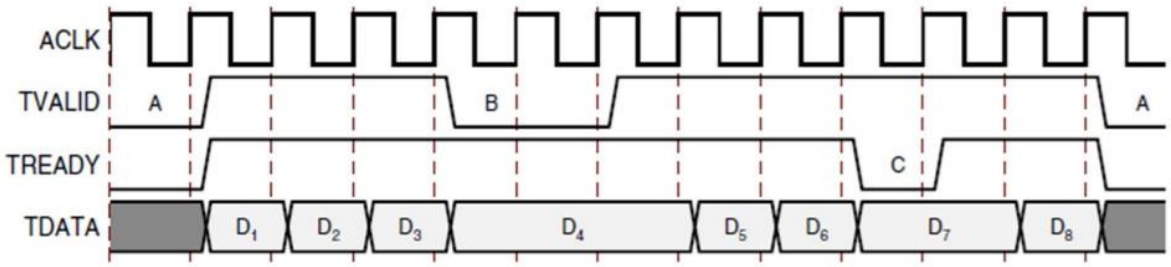
All subsystems are controlled by the Lab CTRL module. The Control Module manages the relevant module's start and ready signals, as well as whether the following module is ready to start. Lab CTRL also separates the data in the window into frames and makes it 50% overlap at this point in the project. As a result, accuracy in voice recognition improves. It is possible to investigate how the overlap rate impacts accuracy by altering it here.

To test Lab CTRL, we need both the ADC and Window modules. The ADC begins sampling the data after the control module gives the start signal. The ADC's ready signal returns to high when the sampling is completed. We used sinus signals of various frequencies to test the PCB, window, ADC, debug, and control modules that had been constructed so far as hardware. Then, in Matlab, we drew each frame and computed the SNR

BURAYA KADAR RAPOR DEGISTIRLDII

3.7 FFT Module

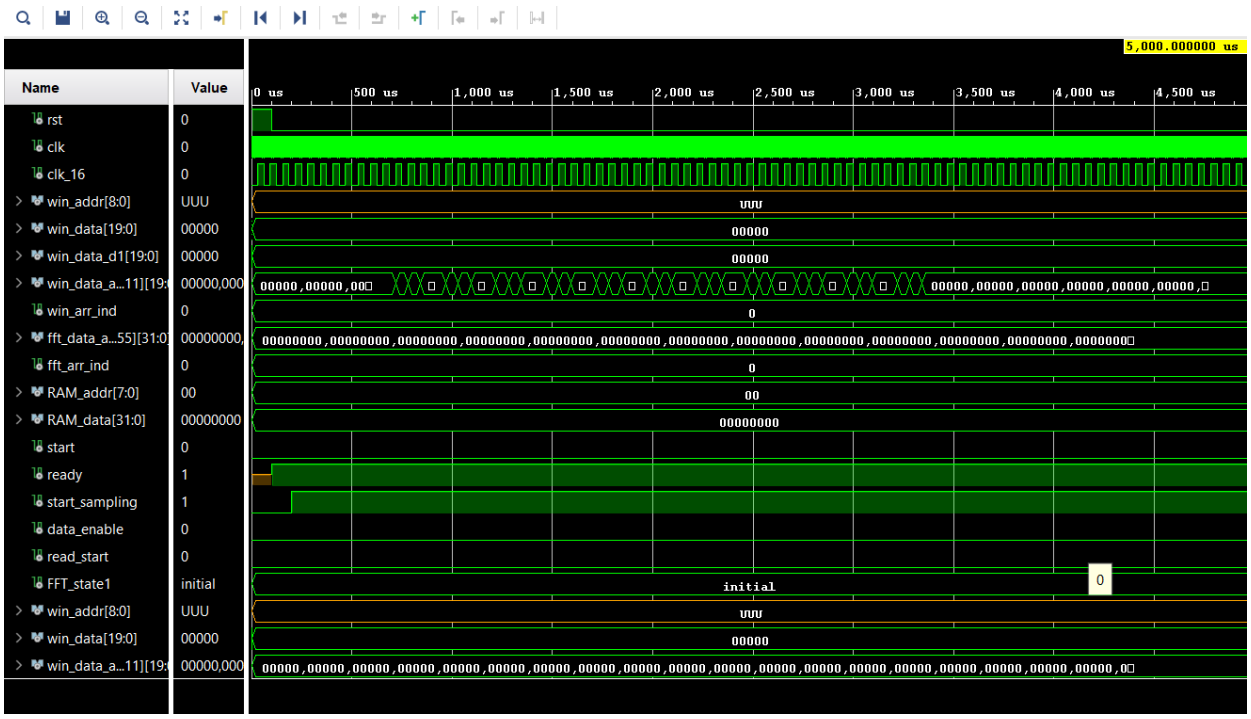
Lab FFT executes a 512-point FFT function by reading 20-bit 512 data samples from the window module from RAM. Then, the real part and the imaginary part of the FFT, whose squared magnitudes we calculate, contain 30 bits each, and we transcribe them to 32 bits and write them back to RAM.



As can be seen from the waveforms given above, the FFT module can receive data when the ready signal is high. However, the data should not change as soon as the ready signal goes to the low state, and as soon as the ready signal returns to the state, we should continue to send the data from where we left off. But we don't know when the ready signal will go to a low state. The FFT module cannot operate and may be overloaded, so the ready signal may switch to low. Unfortunately, it is not possible for us to foresee this scenario. This is why the FFT assignment is the most difficult stage of the project.

In order to solve this problem, we first created a window array signal instead of sending data directly from the window to the FFT. We write the data in the window to this array. Then we send the data to the FFT module. When the ready signal goes low, we can know which index of the array we are in, and we can know from where to start sending data again. Since we write this stage outside of the process, we can perform it instantly without being tied to the clock.

In order to test the FFT module, we created an input file using Matlab. Then we converted this file to coe format and embed it into the ROM. The FFT module reads the ROM and takes the FFT of 512 samples. Then the test bench Lab reads the data from the RAM of the FFT and writes an output text file.



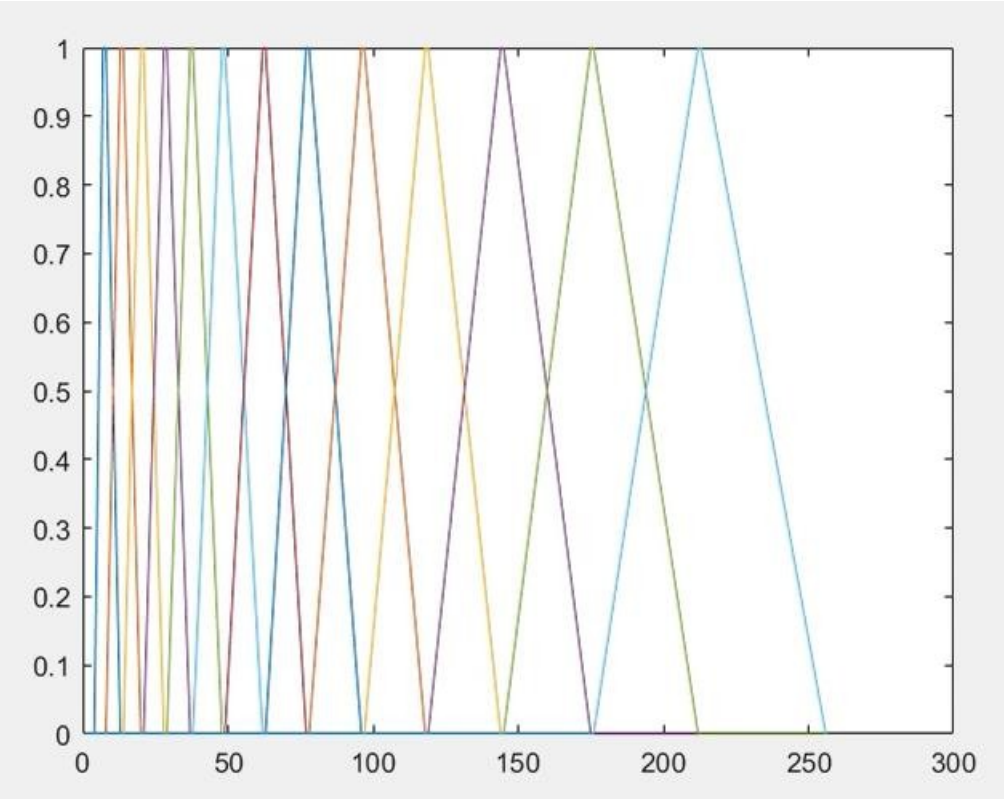
In the hardware phase, we give a sinus signal from the signal generator to the FPGA. We create the signal with the same characteristics in Matlab and calculate the FFT. To compare the FFT calculation of FPGA and Matlab, we plot the graphs of both according to the given signal.

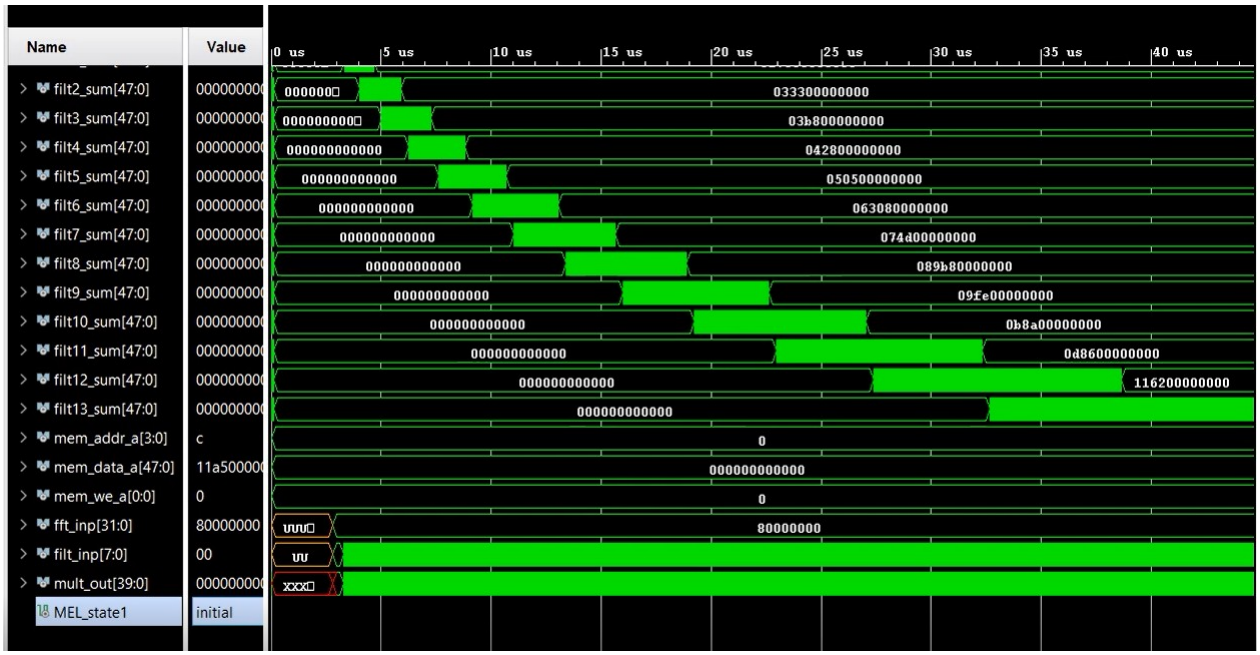
3.8 MEL Module

Lab MEL will read 256 data from the RAM memory of the FFT module and the output of each filter will be used to calculate the band energy. The frequencies of MEL filters vary between 100Hz and 4 kHz and the filters are triangular. After the band energies are calculated, the N-bit address should be saved in the RAM containing you. Here N denotes the number of filters and we use 13 MEL filters.

In order to determine MEL filters, we did a source search and created 13 MEL filters using the necessary algorithms in Matlab. We then wrote these filters into the VHDL code. It could also be embedded in ROM, but we preferred to write it in code.

You can see the MEL filters in the graph we plotted in Matlab.





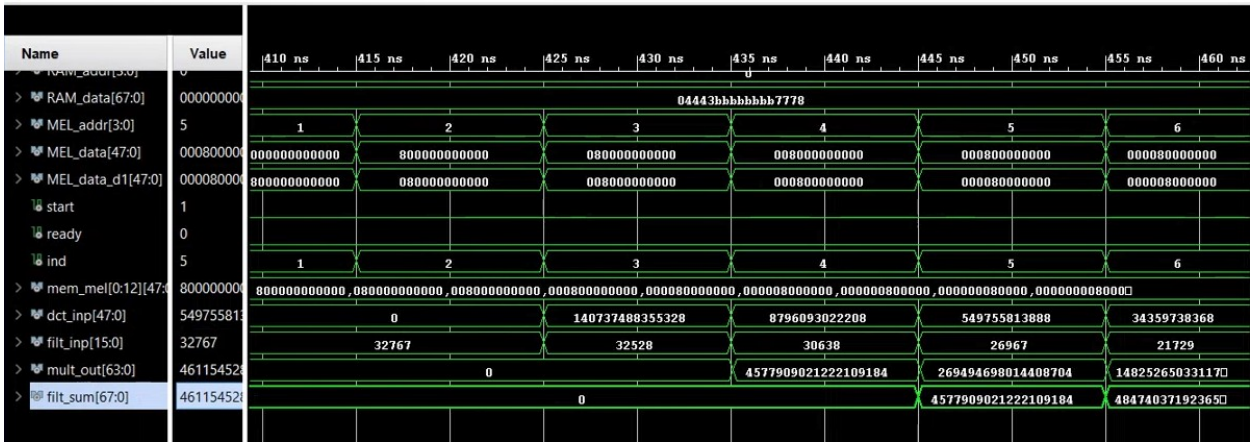
3.9 DCT Module

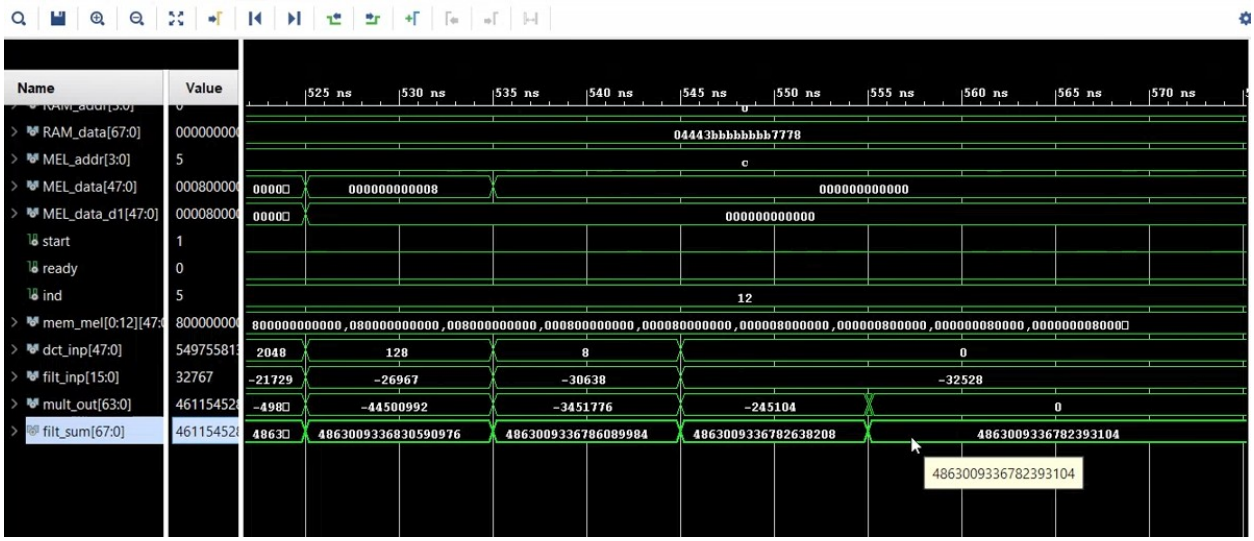
The DCT module will read the energy data from the memory of the MEL and store the results in the dual-port RAM memory again by applying the discrete cosine transform. As stated in the

assignment, the DCT coefficients should be 16 bits and the output is 64 bits because the energy data from the MEL is 48 bits.

The n value in the DCT formula indicates the number of filters and since we have 13 MEL filters, we create the DCTcoefficients accordingly. After creating the DCT coefficients in Matlab, we copied these coefficients to an excel file. Then we converted the coefficients to be hexadecimal by using the DEC2HEX function in excel. Each row you see below represents the DCT coefficients that will be multiplied for 1 MEL filter. 1 MEL filter is summed by multiplying by column 1, column 2 … column 13, respectively, and the result is written to RAM.

We embedded these DCT coefficients in the VHDL code, and as you can see in the simulation, the result of the sum of the times of each MEL filter by the DCT coefficients is the same as the result in our excel.

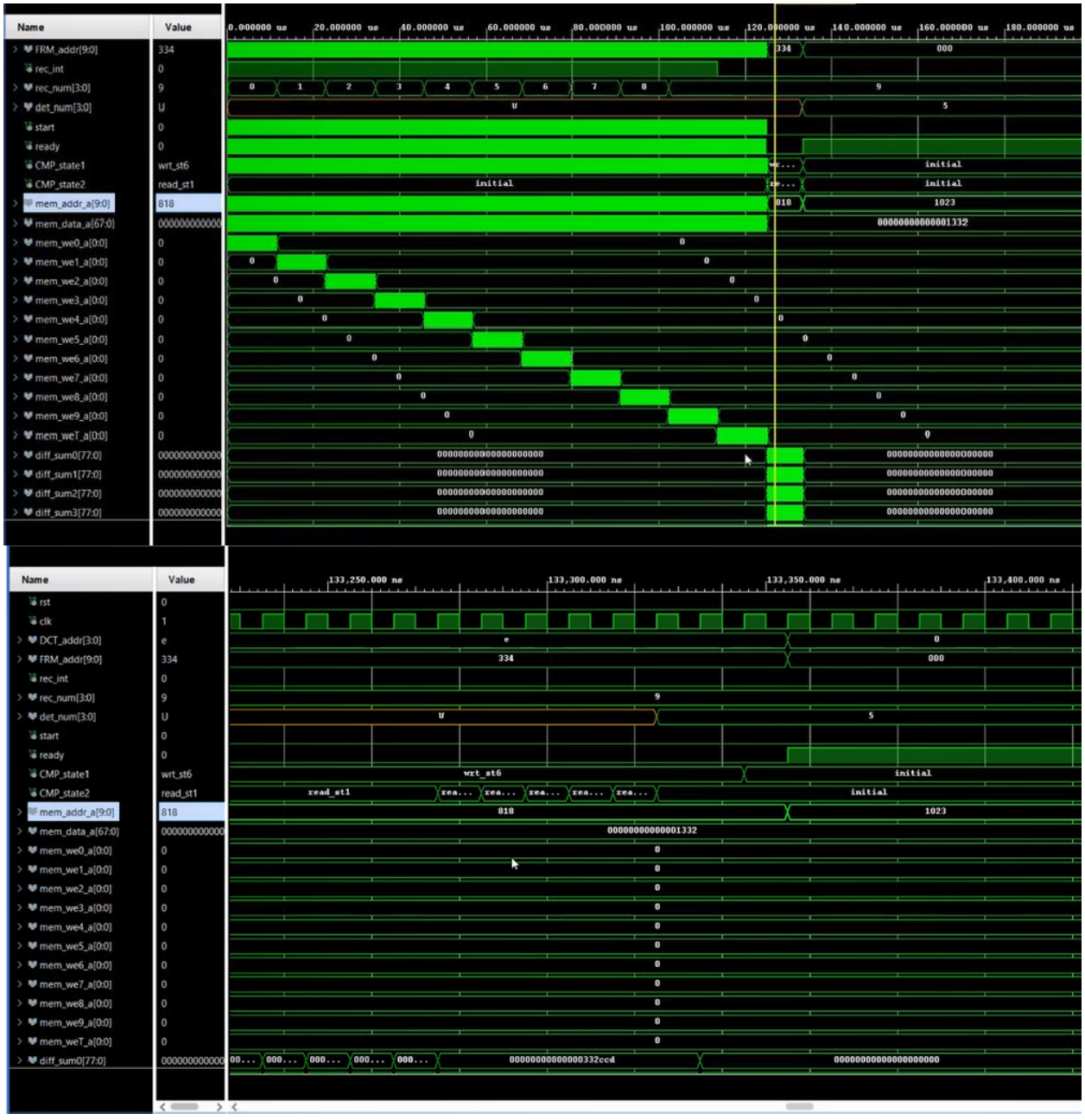




3.10 Compare Module

At the last stage of the project, we implement the Compare module. This module has two modes, record and compare, namely test. In Record mode, we actually record the numbers to be spoken. These numbers are 4 bits, so decimal from 0 to 9. We can say that we create a dataset by saying the numbers one by one to the microphone on the PCB. Then, when we put the compare module in test mode, we say a number and we expect the system to recognize this number and display the correct result on the segment display.

In order to test the Compare module in simulation, we generated random data for each number in Matlab. Since there will be 10 number data and 1 test data, we have created a total of 11 memories. We saved the number of data we generated to these memories. Then we took the data of one number and saved it in the test memory. In this way, we can see if we can detect the correct number as a result of the simulation.

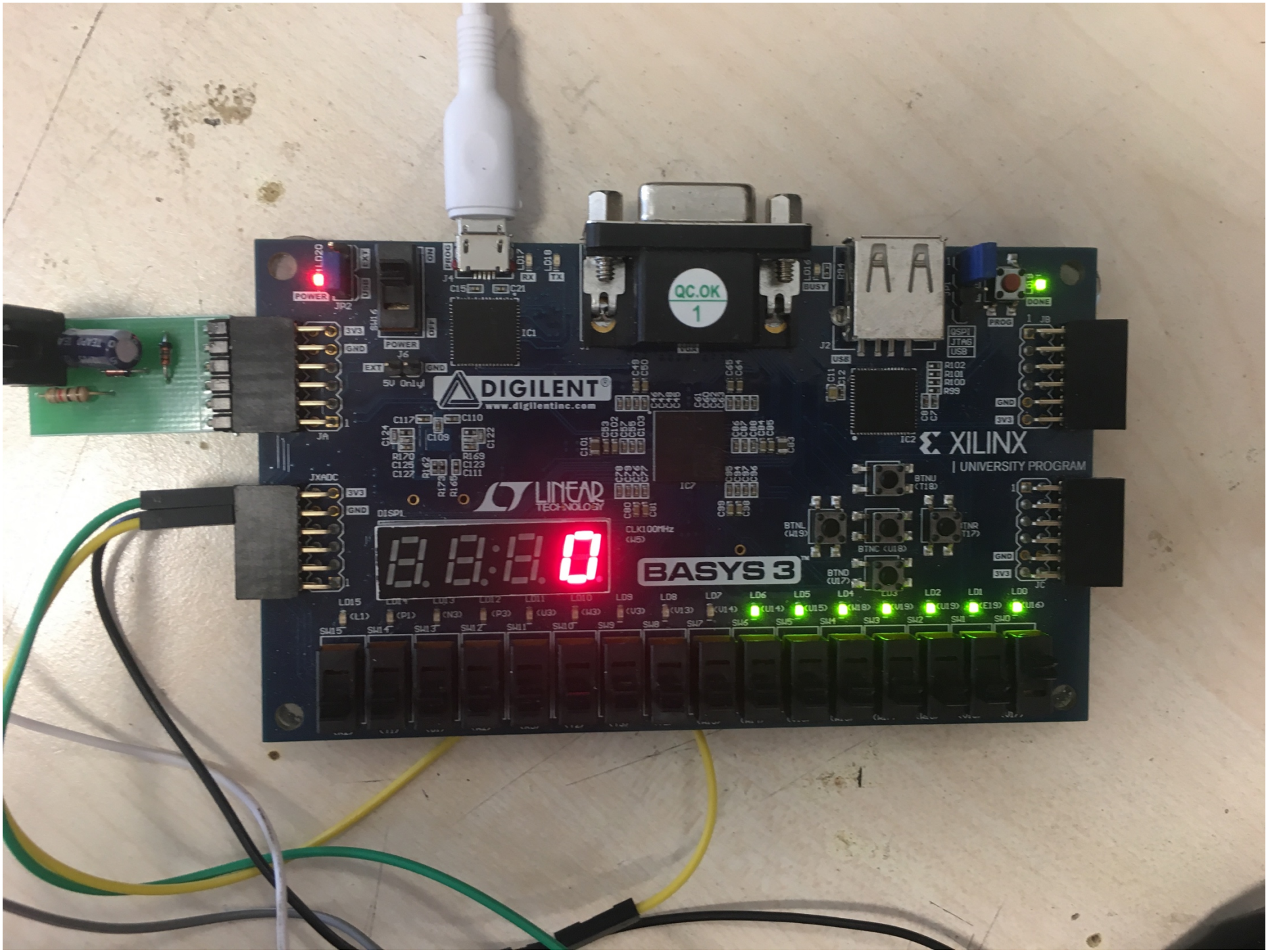


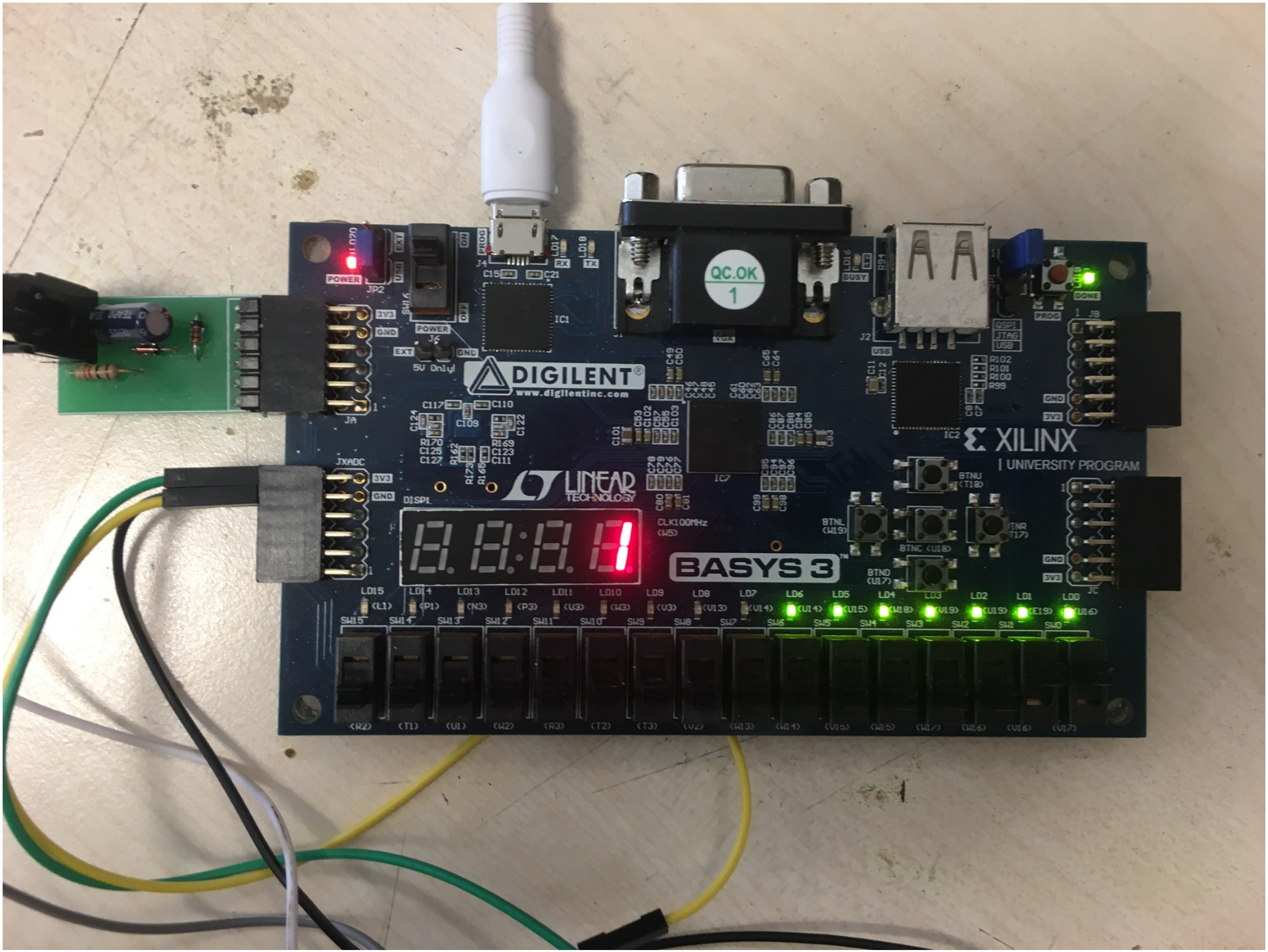
Burdan sonrasi da degisti

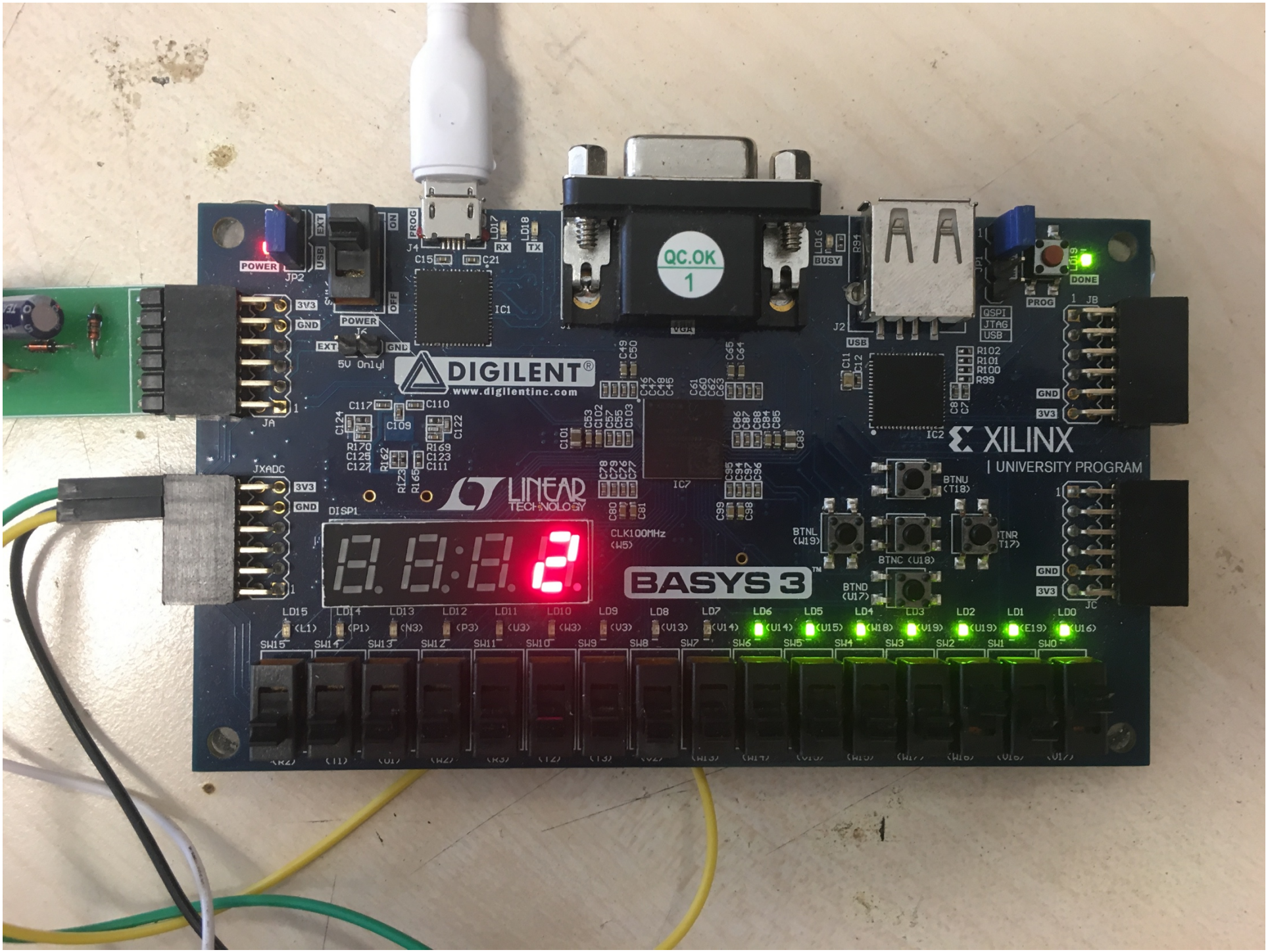
4 DISCUSSION AND CONCLUSION

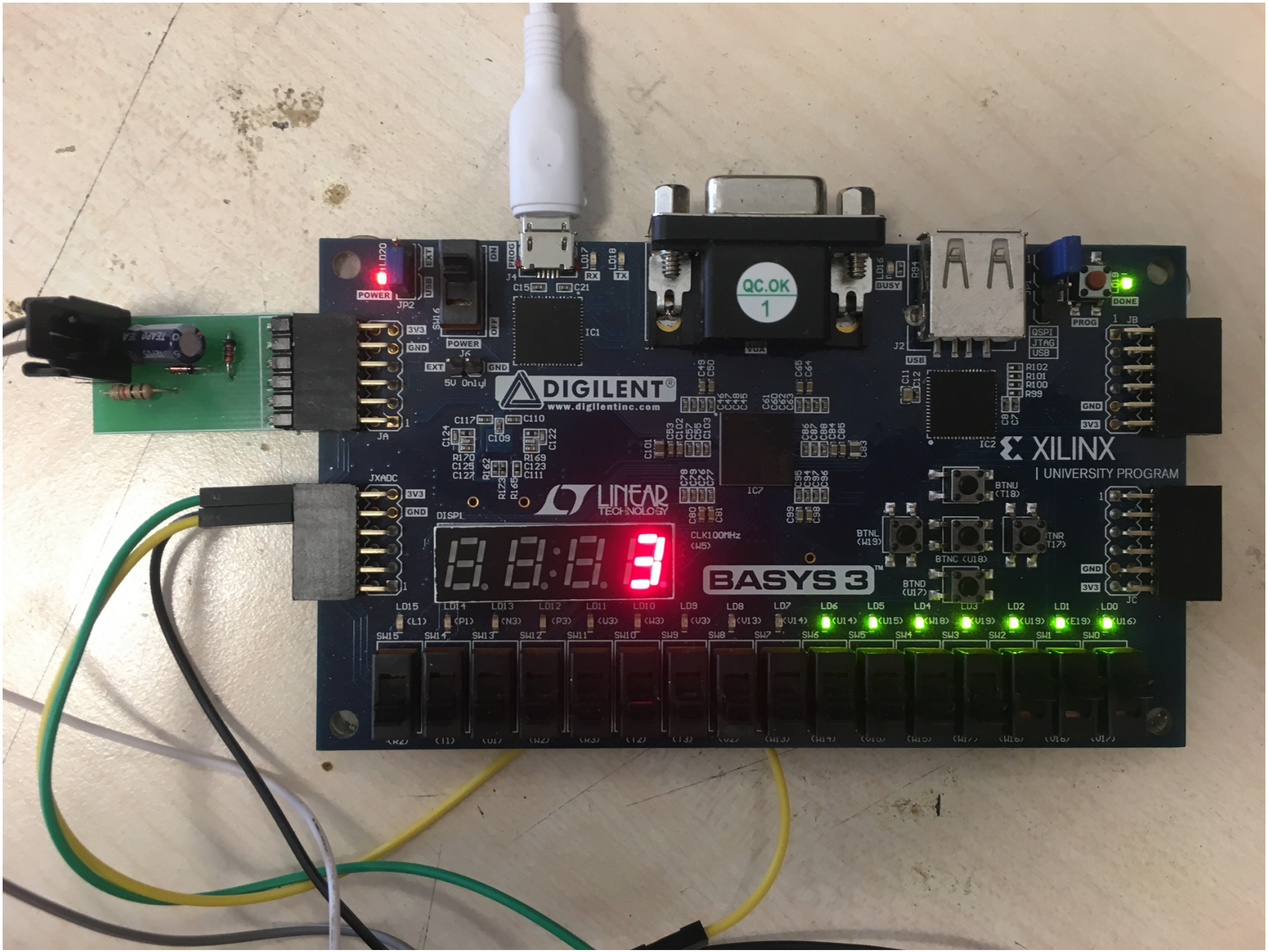
Throughout the semester, we were able to complete all of the experiments on time. We completed all of the assigned tasks and met all of the lab's requirements. All additional laboratories are utilized in the final demonstration, with the exception of Lab - DEBUG and Lab - MATLAB, as their specifics are discussed throughout the report. We were able to show the right values that were told to the system with an accuracy of 80%, compared to 90% in our Lab - MATLAB. This change is to be expected since using an FPGA board as hardware with a PCB and other components increases the risks of mistake and noise.

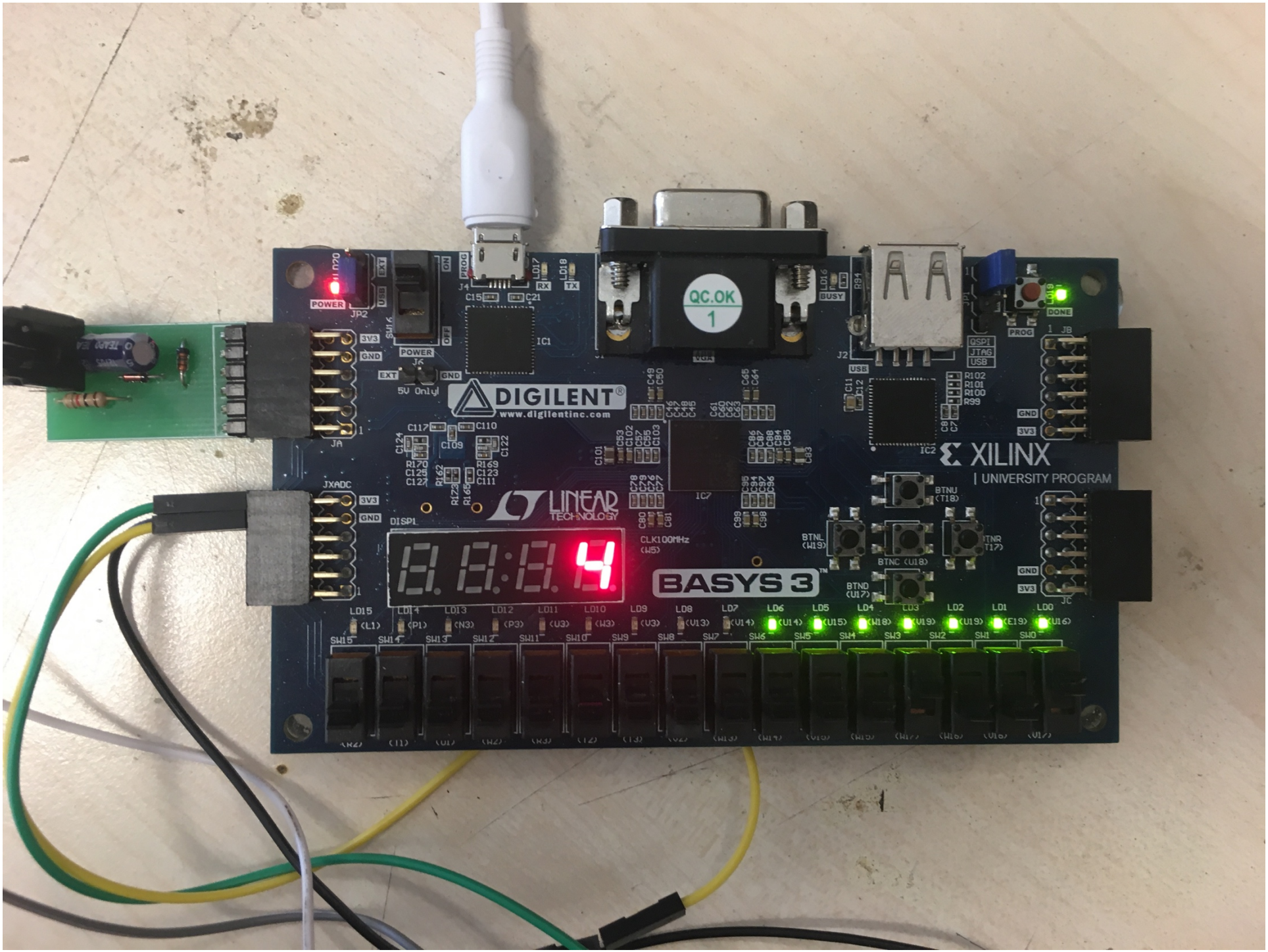
Several significant difficulties arose throughout the development of our system. Making clocks function synchronously and asynchronously is the most challenging. The use of many clocks and attempting to read and write data while considering these clocks was extremely difficult. By adding suitable delays, we were able to make them operate effectively without losing any data between the blocks during the process. Another occurred during the PCB design and execution. It was difficult to draw an acceptable circuit for the PCB and calculate needed values at first. We were able to solve the circuit's calculation and design challenges and move on to the project after conducting extensive study on Sallen - Key Low - Pass Filters.

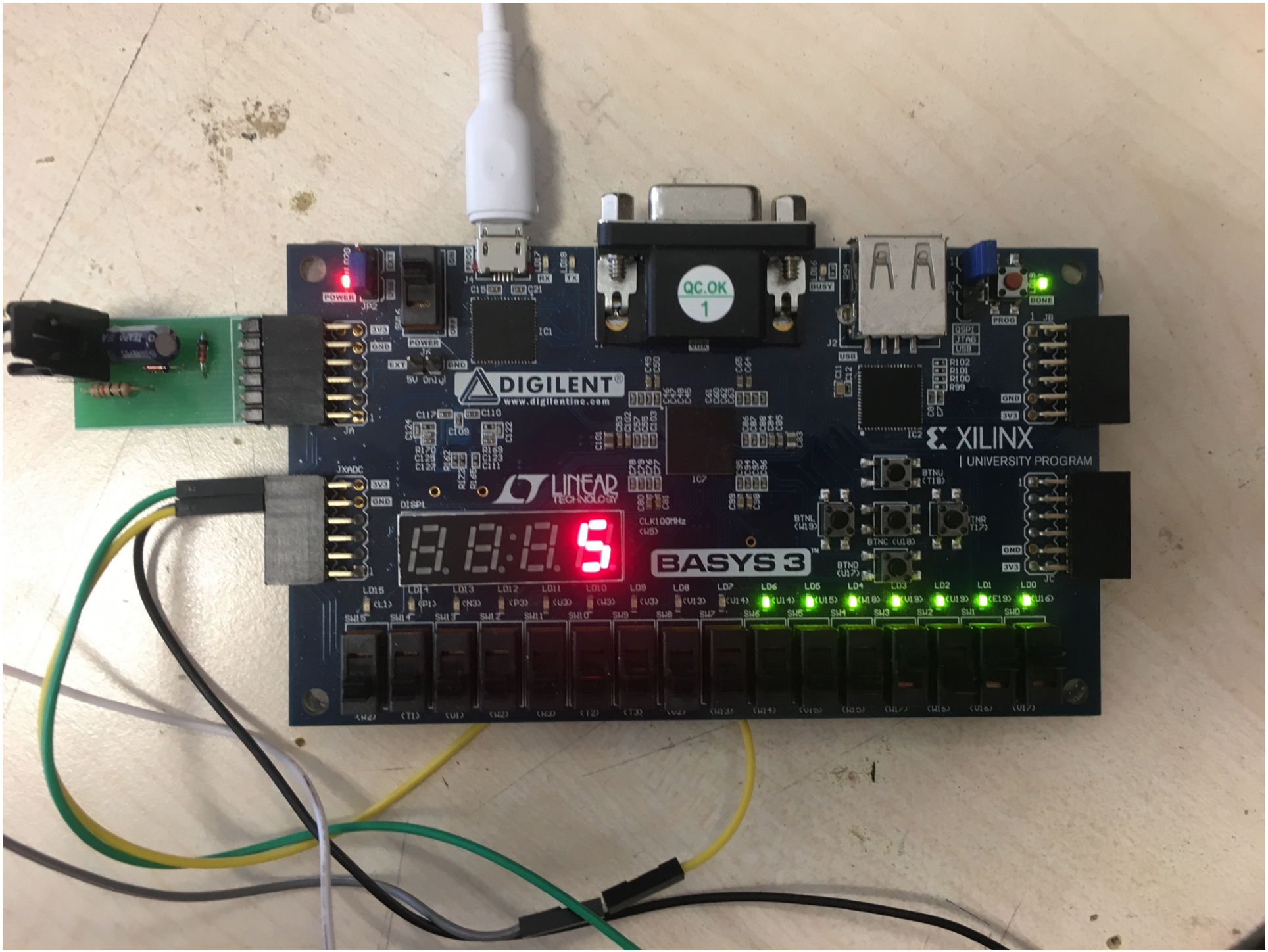

DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
DISP1
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
JA
JB
JC
JXADC
JP2
J2
J4
J6
IC1
IC2
IC7
DONE
PROG
BUSY
5V Only
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0

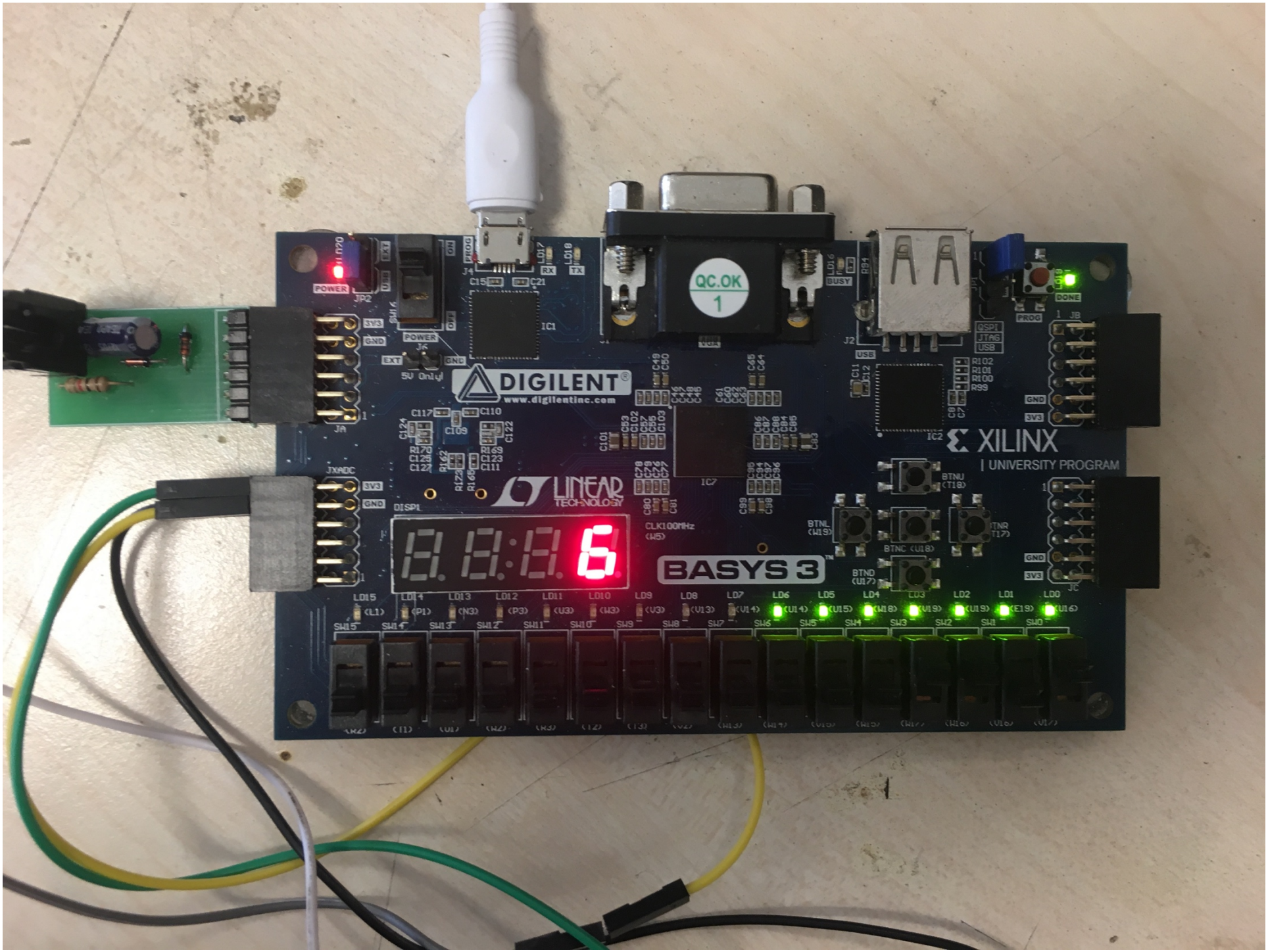

DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
DISPL
JXADC
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTND (U17)
BTNR (T17)
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0
(R2)
(T1)
(U1)
(W2)
(R3)
(T2)
(T3)
(V2)
(W13)
(W14)
(V15)
(W15)
(W17)
(W16)
(V16)
(V17)

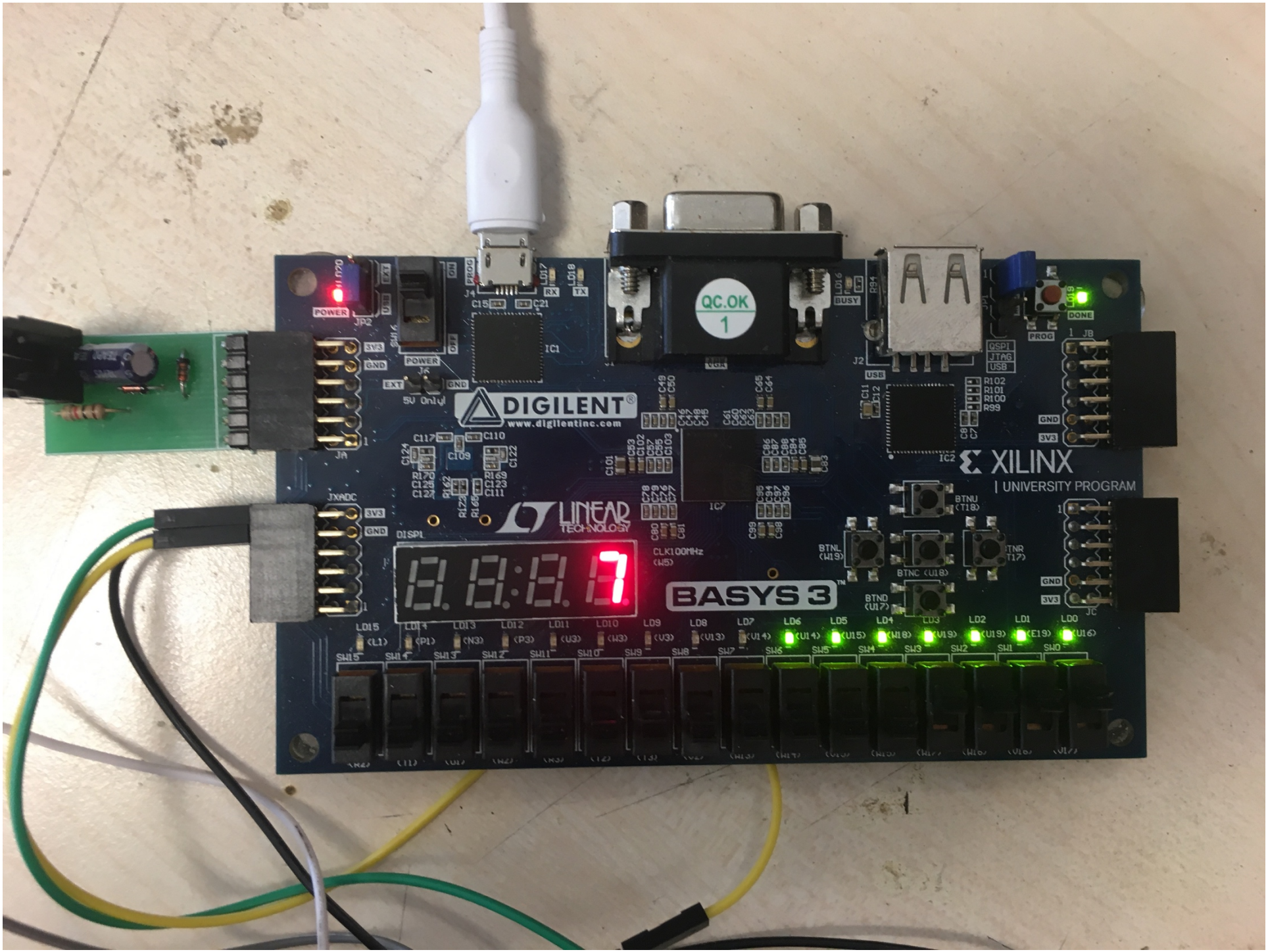

DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
JP2
5V Only!
DISP1
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTND (U17)
JXADC
JA
JB
JC
QSPI
JTAG
USB
PROG
DONE
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)

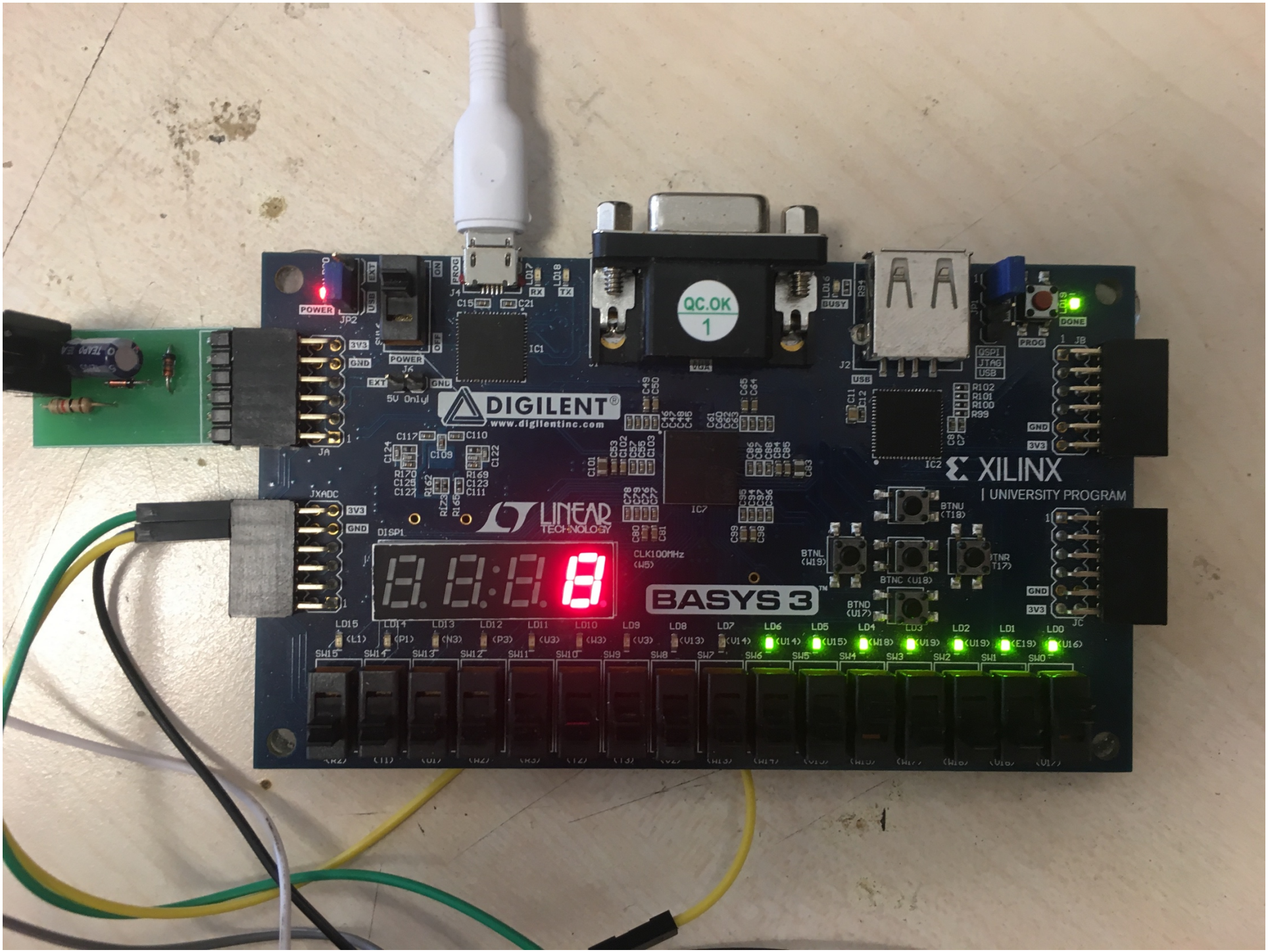

DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
JP2
SW16
J4
IC1
IC2
IC7
J2
JA
JXADC
DISP1
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
PROG
DONE
QSPI
JTAG
USB
JB
JC
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0

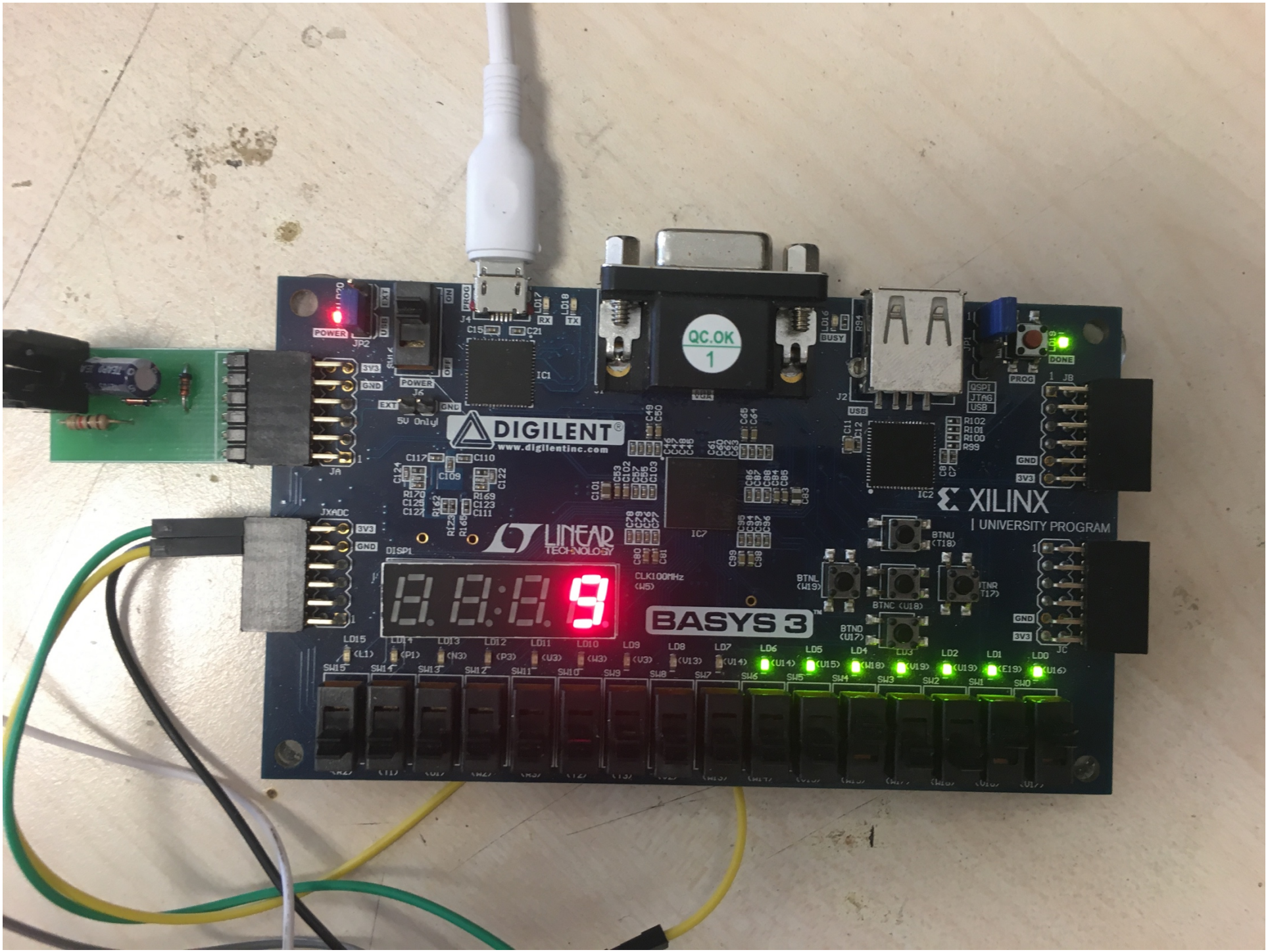

DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
JP2
SW16
PROG
J4
LD17
LD18
RX
TX
C15
C21
IC1
VGA
J2
USB
BUSY
LD16
R94
LD19
DONE
PROG
JB
GSPI
JTAG
USB
R102
R101
R100
R99
C11
C12
IC2
C8
C7
GND
3V3
JA
3V3
GND
EXT
5V Only
J6
POWER
OFF
ON
EXT
USB
JXADC
DISP1
C117
C110
C109
C124
C122
R170
C125
C127
R162
R123
R165
R169
C123
C111
C49
C50
C46
C48
C45
C47
C61
C62
C60
C63
C65
C64
C53
C102
C55
C103
C101
C86
C87
C84
C85
C83
C78
C79
C76
C77
C80
C81
IC7
C95
C94
C97
C96
C99
C98
CLK100MHz
(W5)
BTNU
(T18)
BTNL
(W19)
BTNC (U18)
BTNR
(T17)
BTND
(U17)
JC
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0


QC.OK
1
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
POWER
PROG
DONE
BUSY
DISP1
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR (T17)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
JXADC
5V Only!


QC.OK
1
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNR (T17)
BTNC (U18)
BTND (U17)
LD15 LD14 LD13 LD12 LD11 LD10 LD9 LD8 LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW15 SW14 SW13 SW12 SW11 SW10 SW9 SW8 SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
POWER
PROG
DONE
QSPI
JTAG
USB
5V Only
JA
JB
JC
JXADC


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
PROG
DONE
BUSY
QSPI
JTAG
USB
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTNR T17)
BTND (U17)
CLK100MHz (W5)
DISPL
JXADC
JA
JB
JC
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (V15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
JP2
JA
JXADC
DISP1
CLK100MHz
(W5)
BTNU (T18)
BTNL (W19)
BTNC (U18)
BTND (U17)
BTNR (T17)
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
1
POWER
JP2
J4
IC1
IC2
IC7
JA
JXADC
DISP1
CLK100MHz
(W5)
BTNU
(T18)
BTNL
(W19)
BTNC (U18)
BTNR
T17)
BTND
(U17)
JB
JC
PROG
DONE
USB
QSPI
JTAG
USB
BUSY
5V Only
POWER
J6
EXT
GND
3V3
LD15 (L1)
LD14 (P1)
LD13 (N3)
LD12 (P3)
LD11 (U3)
LD10 (W3)
LD9 (V3)
LD8 (V13)
LD7 (V14)
LD6 (U14)
LD5 (U15)
LD4 (W18)
LD3 (V19)
LD2 (U19)
LD1 (E19)
LD0 (U16)
SW15
SW14
SW13
SW12
SW11
SW10
SW9
SW8
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0