Assignment #1

Problem 1

1. What are the standard clock frequencies used in the microcontroller core of your Arduino and your ESP32 ? What can you tell about the power of each board?

The Arduino is powered by the nRF52840 microcontroller and has a clock speed of 64 MHz, while the ESP32 has at the core the ESP32-D0WD chip which has two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz.

2. Write about the various sensors in the Arduino in the following format. Fill up the table.

S.No.	Sensor	Part Number	Function
1	9 axis inertial sensor	LSM9DS1	3D Accelerometer, 3D Gyroscope, 3D Magnetometer
2	Microphone	MP34DT05	detecting acoustic waves
3	Gesture, Light, Proximity Sensor	APDS9960	Ambient Light and RGB Color Sensing, Proximity Sensing, and Gesture Detection in an Optical Module
4	Barometric Pressure Sensor	LPS22HB	absolute pressure sensor which functions as a digital output barometer
5	Temperature & Humidity Sensor	HTS221	16-bit humidity and temperature output data

3. What is the largest acceleration that the sensor chip (LSM9DS1) can measure?

Up to 16 G-forces which is about 157 m/s^2

4. Select two of the sensors on your Arduino. For each sensor, write a program on your board to receive data and print it to the terminal.

Zipped LPS22HB_Pressure.ino, LSM9DS1_Gyroscope.ino, and p1_4.mov

5. Mention 4 key feature/performance differences between ESP32 and Arduino. Recall what was discussed in the discussion. (Your answer should be more than 5 sentences)

The Arduino offers 5 built in sensors that we can interface with. While the ESP32 does offer higher cpu clock speeds, greater flash memory and SRAM. Additionally the ESP32 can be interfaced with the same arduino ide, but it can also be coded with python. The ESP32 also has more I/O pins as well as can operate at a higher voltage.

6. What is the CPU architecture used in the Arduino? What will happen if you draw too much power from the Arduino? (Your answer should not be more than 3 sentences)

The CPU is a 32-bit ARM Cortex™-M4 which uses Arms Cortex M Profile architecture. Drawing too much power can heat up the arduino, which can cause chip failure and burn out the device.

Problem 2

1. Using your ESP32, generate a digital waveform with a 2 second period and a 25% duty cycle. Apply your waveform to control the Built_In LED of the board. (Use the delay function for this part)

Zipped ESP32_with_delay.ino

2. For part 1 name two alternatives that can be used instead of the delay() function.

Alternatively the millis() and micros() functions in a loop can be used instead of delay().

3. Rewrite your waveform function and this time make sure you are not using the delay() function.

Zipped ESP32_without_delay.ino, and p2_3.mov

4. Following question 3, tell us why you should not always use the delay function? How do the other methods you mentioned handle the issue with the delay() function.

The delay function halts everything for however long it is set to, and while waiting nothing else can be done. This wastes time and is pretty inefficient. Using the millis() and micros() functions in a loop.

5. Going back to your Arduino, now use your timing technique to read values from both your sensors in the same program. The two sensors could be the ones you worked with in problem 1. Update and merge your code so that you would read Sensor A every 5 seconds and Sensor B every second.

Zipped LPS22HB_Pressure_without_delay.ino, LSM9DS1_Gyroscope_without_delay.ino, p2_5.mov

Problem 3

1. In no more than 5 sentences, explain why using interrupts are so important in programming microcontrollers. Give us two examples, when a programmer uses interrupts.

Interrupts are important as they can stop the main program from running to do something that might be of importance. Without an interrupt the cpu would have to waste time polling hardware for information, instead hardware can briefly interrupt and give the cpu the information they needed to share and then the cpu can go back to what it was doing. There can be hardware interrupts like user input or software interrupts like a program needing resources unexpectedly.

2. Mention two ways to make your interrupt routines more reliable in ESP32.

On the ESP32 all GPIO pins can be configured as interrupts. You can also use the function attachInterrupt() to set up interrupts based on which pin you choose.

3. Using your ESP32, run the final version of your PWM code from the previous problem. The pin that outputs your digital wave should be D2 (BUILT_IN LED). Now, use a jumper wire to connect D2 to pin D13 on your board. Using interrupts now, can you calculate the duty cycle (D) and period (T) of your wave through D13? Write a code to print this information in your terminal window.

Zipped ESP32_without_delay_interrupt.ino, and p3_3.mov

(ALL VIDEOS ADDED TO DRIVE FOLDER 205868607_Johnson)
(ALL CODE ZIPPED WITH THIS PDF)