Smartwatch Developed Using The PIC18F4620 Microcontroller

I am **Yassin Khadrawy**, a student at the Communication and Electronics Department, Faculty of Engineering, Alexandria University.

Today, I am going to talk about my latest project, which is a **smartwatch developed using the PIC18F4620 microcontroller.**

In this project, I utilized various peripherals of the PIC18F4620 microcontroller, including **communication peripherals**, **timer peripherals**, **PWM peripherals**, **ADC peripherals**, and **GPIO peripherals**.

The components I incorporated in this project are:

- **Rain Sensor**: For detecting rain.
- **Temperature Sensor**: To measure ambient temperature.
- **Pressure Sensor**: To monitor atmospheric pressure.
- **GPS Module**: For location tracking.
- **LDR (Light Dependent Resistor)**: To measure light intensity.
- **LCD**: To display information such as time, date, or sensor data.
- **LED**: For visual indications.
- **Buttons**: For user interaction and control.
- **Resistors**: As part of the supporting circuitry.

These components work together to create a functional smartwatch with enhanced features.

Buttons

1. **Start Button**: This button allows you to adjust the clock when it is released. Once pressed, the watch will start, and you will no longer be able to adjust the time.
2. **Hours Button**: This button allows you to adjust the hours, but only when the Start button is released.
3. **Minutes Button**: This button allows you to adjust the minutes, but only when the Start button is released.
4. **Charging Button**: This button must be pressed when the battery level reaches 20%. It charges the watch back to 100% before it is powered off. Once powered off, the watch cannot be turned back on unless charged again.

Sensors

1. **Temperature Sensor(LM45)**: This sensor measures the ambient temperature in Celsius, wherever you are.
2. **Pressure Sensor(MPX4115)**: This sensor measures the atmospheric pressure in kPa (kilopascals).
3. **Rain Sensor**: This sensor detects whether it is raining or not. In Proteus, it is simulated by using a logic state. When the value is 1, it indicates that it is raining, and when the value is 0, it means there is no rain.
4. **LDR (Light Dependent Resistor)**: This sensor detects the amount of light around the watch. When the light intensity is low, a red LED will turn on. The LED's brightness will increase or decrease based on the LDR's resistance, which changes with the light intensity.
5. **GPS**: The GPS module allows you to integrate and test location-based systems. It calculates the latitude, longitude, altitude, and provides real-time date and time. All this data is received by the PIC microcontroller and displayed on a virtual terminal.

Timer0 (Peripheral at PIC18F4620)

Timer0 is used to calculate time intervals accurately, specifically to count seconds. At each loop, the timer increments by one second. To ensure correct timekeeping, I used a **Timer0 calculator** to determine the appropriate values for the timer settings.

- **Prescaler**: 1:32
- **TMR0 Preload**: 3036
- **Actual Interrupt Time**: 1 second
- **Timer Mode**: 8-bit mode

The **Prescaler** of 1:32 is used to slow down the timer, allowing the timer to increment in a way that corresponds to real-world time. The **TMR0 Preload** value of 3036 ensures that the timer generates an interrupt at a precise interval, allowing me to count one second accurately.

This configuration allows the watch to maintain accurate time by incrementing each second based on the timer's interrupt.

USART (Peripheral at PIC18F4620)

I use **USART (Universal Synchronous Asynchronous Receiver Transmitter)** to receive and transmit data via the GPS module. The USART is configured with the **BAUDRATE_ASYN_8BIT_LOW_SPEED** setting, which allows low-speed asynchronous communication.

- **Transmit and Receive**: Both transmission and reception are enabled for USART.
- **Interrupts**: Interrupts are enabled for both the transmit and receive operations to handle data asynchronously.

In my application, the **GPS module** only sends data to the PIC microcontroller. However, you can also enable two-way communication (both receive and transmit) by configuring the **EUSART_TxDefaultInterruptHandler** function in the USART configuration within the **mcal_layer_init**.

This allows the microcontroller to not only receive data from the GPS module but also send data if needed for future functionalities (such as sending data to a server or another device).

ADC (Analog-to-Digital Converter) - Peripheral at PIC18F4620

In my project, I utilize the **Analog-to-Digital Converter (ADC)** to convert analog signals from sensors into digital values that the microcontroller can process. This allows the sensors to provide accurate readings.

- **Number of Channels Used**: 3 analog channels
  - **AN0**: Connected to the **Temperature Sensor** to measure ambient temperature.
  - **AN1**: Connected to the **Pressure Sensor** to measure atmospheric pressure.
  - **AN2**: Connected to the **LDR (Light Dependent Resistor)** to detect light intensity.

The ADC is crucial in enabling these sensors to interact with the microcontroller, as it ensures the analog signals are correctly interpreted and used in the application's logic.

GPIO (General Purpose Input/Output) - Peripheral at PIC18F4620

In my project, I use **GPIO (General Purpose Input/Output)** to initialize all the microcontroller's pins as either input or output. The GPIO functionality relies on three key registers:

1. **TRIS Register**: Configures each pin as either an input or an output.
2. **PORT Register**: Reads the voltage level on the pin (5V or 0V).
3. **LAT Register**: Writes the desired voltage level (5V or 0V) to the pin.

Applications in the Project

- **Direct Uses**:
  - Initializing the **buttons** and **LED** as either input or output pins.
- **Indirect Uses**:
  - Supporting other drivers such as **ADC**, **USART**, and other peripherals through GPIO configuration.

Additionally, GPIO is used in the **ECU Layer** to set up various components and peripherals needed in the project. By configuring the pins correctly, GPIO serves as the foundation for interfacing with sensors, modules, and other hardware.

PWM (Pulse Width Modulation) - Peripheral at PIC18F4620

In my project, I use **PWM (Pulse Width Modulation)** to control the behavior of the LED indirectly through the **LDR (Light Dependent Resistor)**.

- **Functionality**:
  The PWM signal adjusts the LED's brightness based on the light intensity detected by the LDR:
  - When the light intensity on the LDR increases, the LED brightness decreases.
  - When the light intensity on the LDR decreases, the LED brightness increases.

This relationship ensures dynamic control of the LED brightness, making it responsive to the surrounding light conditions. The PWM duty cycle is varied accordingly to achieve this effect.

This concludes the documentation of my latest project, the **Smart Watch**. I hope this documentation helps you understand the various components, peripherals, and functionality of the project. Thank you for your interest and time in exploring my work!

Eng:**Yassin Khadrawy**