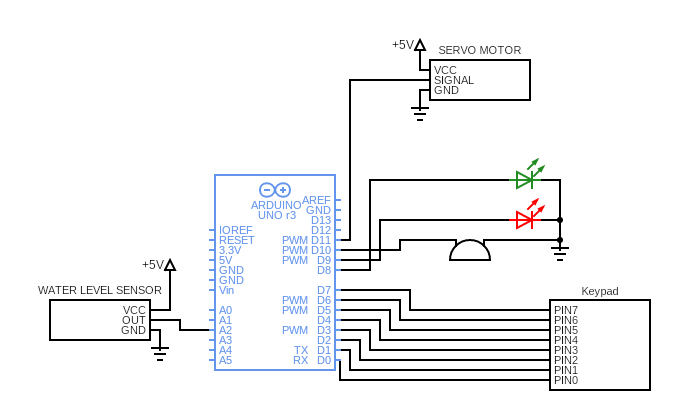
**FLOOD DETECTION & RESCUE**

**Circuit diagram:**

****

**Working of components**

**Arduino UNO Board:**

The user can get started by connecting the Uno to a computer with the USB cable or by powering it with an AC / DC adapter or battery. The Uno can be programmed with Arduino Software (Integrated Development Environment). Arduino Uno features 14 digital input / output pins (six of which can be used as PWM outputs), six analog inputs, and a 16MHz quartz crystal. Uno also includes a USB connection, a power jack, an In- Circuit Serial Programming (ICSP) header, and a reset button. This Arduino MCU board contains everything the user needs to support the MCU.

**Servo Motor:**

The servo motor is connected to the D8 pin of Arduino. A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer. Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating. Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer’s angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided. At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

**4x4 Matrix keypad:**

The matrix keypad is connected to the pins D0 to D7 of Arduino. This 4x4 matrix keypad has 16 built-in pushbutton contacts connected to row and column lines. A microcontroller can scan these lines for a button-pressed state. In the keypad library, the Propeller sets all the column lines to input, and all the row lines to input. Then, it picks a row and sets it high. After that, it checks the column lines one at a time. If the column connection stays low, the button on the row has not been pressed. If it goes high, the microcontroller knows which row (the one it set high), and which column, (the one that was detected high when checked). See the schematic in the "Circuit" section, above, for a visual reference of the keypad layout.

The keypad library supports pretty much any number of rows and columns. So, the program has to tell it our keypad is has 4 rows and 4 columns, which I/O pins the lines are connected to, and what value each button represents. The rows, cols, and values arrays store that information. The rows array will be used to tell the keypad library that the top row is connected to P7, the second row to P6 and so on. Likewise, the cols array lists the leftmost column as connected to P3, the next over connected to P2 and so on. The values array stores the value we want the program to give us for each button press. For example, if the top-left button is pressed, we want the number 1, and if the next one over is pressed, we want the number two. If the top right button is pressed, we want the ASCII code for the 'A' character, which is 65.

Inside the main function, keypad setup gets the number of rows (4), the number of columns (also 4), the rows array, the cols array, and the values array. After that, key = keypad read () will return -1 if no buttons are pressed. If a button is pressed, it will return the value from the array that corresponds to that button. For example, if you press a button on the 3rd row, second column, the keypad read function will return the number 8, which will get stored in the key variable. To display it correctly, an if statement checks for values less than or equal to 9, and displays them with %d, decimal-integer formatting flag. The ASCII codes for '\*', '#', 'A', 'B', 'C', and 'D' are 35, 42, 65, 66, 67, and 68, all of which are above 9, and get displayed with the print statement that uses the %c character formatting flag.

**Water Level Sensor:**

The water level sensor is connected to the A2 pin of Arduino. The working of the water level sensor is pretty straightforward. The series of exposed parallel conductors, together acts as a variable resistor (just like a potentiometer) whose resistance varies according to the water level. The change in resistance corresponds to the distance from the top of the sensor to the surface of the water. The resistance is inversely proportional to the height of the water. The more water the sensor is immersed in, results in better conductivity and will result in a lower resistance. The less water the sensor is immersed in, results in poor conductivity and will result in a higher resistance. The sensor produces an output voltage according to the resistance, which by measuring we can determine the water level.

**LEDs:**

Here we are using two LEDs. Red and green. The red LED is connected to the D5 pin and the green is connected to the D4 pin of Arduino. The light-emitting diode simply, we know as a diode. When the diode is forward biased, then the electrons & holes are moving fast across the junction and they are combined constantly, removing one another out. Soon after the electrons are moving from the n-type to the p-type silicon, it combines with the holes, then it disappears. Hence it makes the complete atom & more stable and it gives the little burst of energy in the form of a tiny packet or photon of light.

**Buzzer:**

The buzzer is connected to the D3 pin of Arduino. It consists of an outside case with two pins to attach it to power and ground. Inside is a piezo element, which consists of a central ceramic disc surrounded by a metal (often bronze) vibration disc. When current is applied to the buzzer it causes the ceramic disk to contract or expand. Changing the This then causes the surrounding disc to vibrate. That’s the sound that you hear. By changing the frequency of the buzzer, the speed of the vibration’s changes, which changes the pitch of the resulting sound.