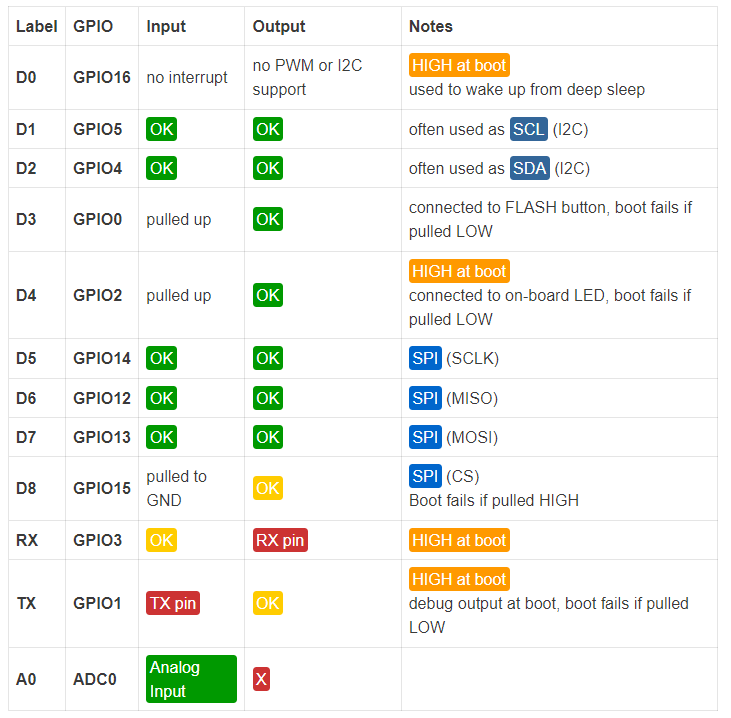
**Schematic Diagram 1.1**

A picture containing text, scoreboard, screenshot

Description automatically generated

**Diagram

Description automatically generated**

****

* + **Power Pins** here are four power pins. **VIN** pin and three **3.3V** pins.
    - **VIN** can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on **VIN** is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the **VIN** pin
    - **3.3V** pins are the output of the onboard voltage regulator and can be used to supply power to external components.
  + **GND** are the ground pins of NodeMCU/ESP8266
  + **I2C Pins** are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.
  + **GPIO Pins** NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts.
  + **ADC Channel** The NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. Testing power supply voltage of VDD3P3 pin and testing input voltage of TOUT pin. However, they cannot be implemented at the same time.
  + **UART Pins** NodeMCU/ESP8266 has 2 UART interfaces (UART0 and UART1) which provide asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.
  + **SPI Pins** NodeMCU/ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs also support the following general-purpose SPI features:
    - 4 timing modes of the SPI format transfer
    - Up to 80 MHz and the divided clocks of 80 MHz
    - Up to 64-Byte FIFO
  + **NSDIO Pins** NodeMCU/ESP8266 features Secure Digital Input/Output Interface (SDIO) which is used to directly interface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.
  + **TPWM Pins** the board has 4 channels of Pulse Width Modulation (PWM). The PWM output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μs to 10000 μs (100 Hz and 1 kHz).
  + **Control Pins** are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    - **RST:** RST pin is used to reset the ESP8266 chip.
    - **WAKE:** Wake pin is used to wake the chip from deep-sleep.
  + Control Pins are used to control the NodeMCU/ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST) and WAKE pin.
    - **EN:** The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.
    - **RST:** RST pin is used to reset the ESP8266 chip.
    - **WAKE:** Wake pin is used to wake the chip from deep-sleep.

Specifications and Features of NodeMCU:

* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT projects

A 32-bit processor with 16-bit instructions powers the ESP8266. Because of the Harvard architecture, instruction memory and data memory are entirely independent.

The Read-Only Memory (ROM) of the ESP8266 contains a first stage boot loader and some library programmes. The remaining code needs to be kept in an external Serial flash memory (provides only serial access to the data – rather than addressing individual bytes, the user reads or writes large contiguous groups of bytes in the address space serially).

The quantity of flash memory that is readily available can differ depending on your ESP8266.

The ESP8266 features a set of GPIO pins (General Purpose Input/output pins) that can be used to “control” external sensors, much like any other microcontroller.

Only 11 of the 17 GPIO pins on our ESP8266 can be utilised (among 17 pins, 6 are used for communication with the on-board flash memory chip). There is also an analogue input (to convert a voltage level into a digital value that can be stored and processed in the ESP8266).

Additionally, it features WIFI connection, allowing you to link your ESP8266 to your WiFi network, access the internet, run a web server, enable smartphone connectivity, etc.

The ability to be programmed like any other microcontroller, notably any Arduino, is another benefit of an ESP8266.

**Learning outcome 1.1**

Through the exercise, we learnt about the pin arrangement, architecture, and specifications of the NodeMCU, as well as the basics of the NodeMCU.

**Aim of Activity 1.2**

To blink the inbuilt LED’s of NodeMCU

**Schematic Diagram 1.2**

|  |
| --- |
|  |
|  |  |

**Program Code 1.2**

void setup() {

  pinMode(LED\_BUILTIN, OUTPUT);

}

void loop() {

  digitalWrite(LED\_BUILTIN, HIGH);

  delay(1000);

  digitalWrite(LED\_BUILTIN, LOW);

  delay(1000);

}

**Results 1.2**

**Led 1                                                       Led 2**

|  |
| --- |
| A picture containing text, indoor  Description automatically generated |
|  | A picture containing text, table  Description automatically generated |

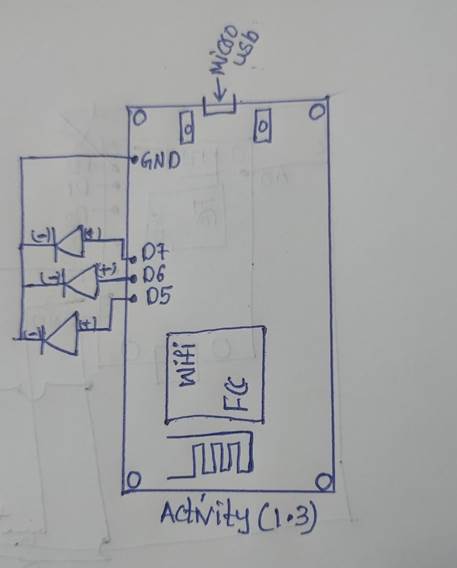
**Learning outcome 1.2**

We wrote a software that blinks both of NodeMCU's internal LEDs and displays the results on the node MCU.

**Activity 1.3**

**Aim:** To blink the External LEDs connected to pin number D5, D6, D7 of NodeMCU

**Schematic Diagram 1.3**



**Program Code 1.3**

int led = 14;

int led2 = 12;

int led3 = 13;

// the setup routine runs once when you press reset:

void setup() {

  // initialize the digital pin as an output.

  pinMode(led, OUTPUT);

  pinMode(led2, OUTPUT);

pinMode(led3, OUTPUT);

}

// the loop routine runs over and over again forever:

void loop() {

  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)

  delay(1000);               // wait for a second

  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW

  delay(1000);

  {digitalWrite(led2, HIGH);

delay(1000);

digitalWrite(led2, LOW);

delay(1000);}

{digitalWrite(led3, HIGH);

delay(1000);

digitalWrite(led3, LOW);

delay(1000);}// wait for a second

}

|  |
| --- |
| A picture containing electronics, adapter  Description automatically generatedA picture containing text, table, indoor, floor  Description automatically generated |
|  | A picture containing text, electronics  Description automatically generated |

**Results 1.3**

**Learning outcome 1.3**

During this exercise, we blinked the leds connected to pins D5, D6, and D7, ran the programme with NodeMCU, and then displayed our results on the board.

**Activity 1.4**

**Aim:** To print the data on Serial monitor of NodeMCU Schematic Diagram

**Program Code 1.4**

void setup() {

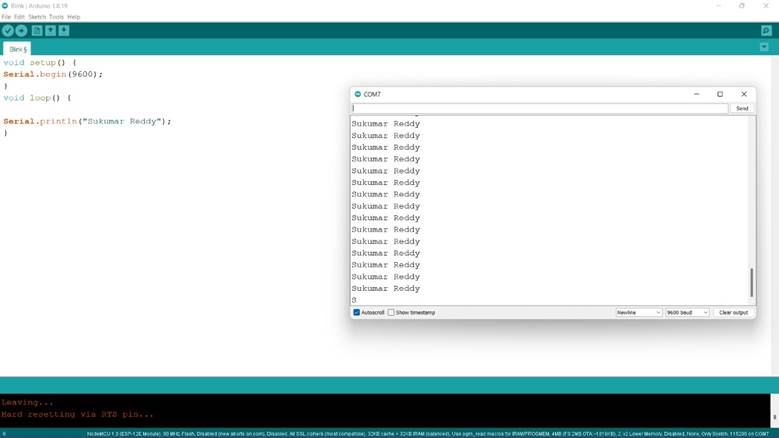
Serial.begin(9600);

}

void loop() {

Serial.println("Sukumar Reddy");

}



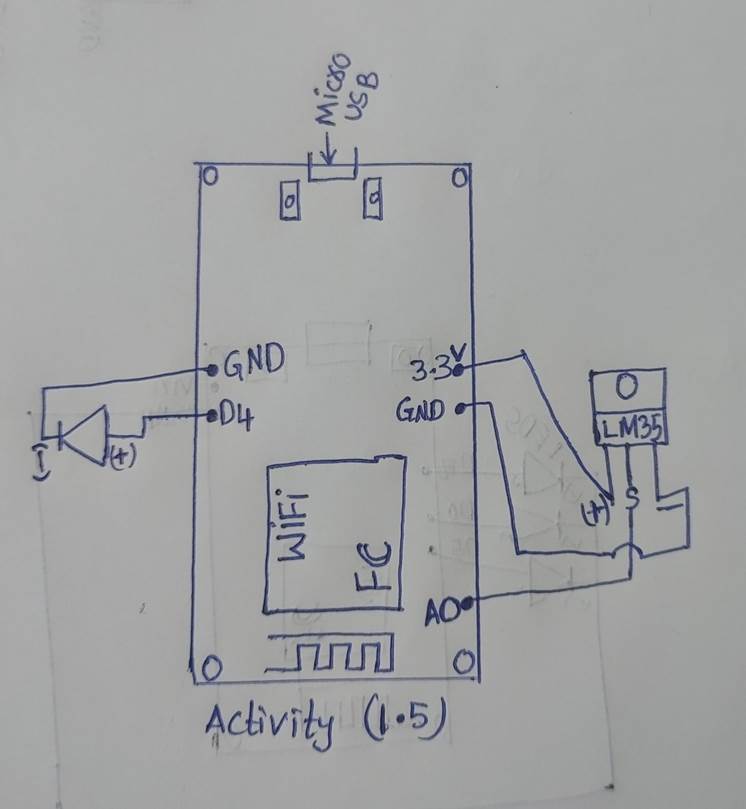
**Results 1.4**

We wrote data on the serial monitor of the Arduino and validated it on the serial monitor throughout this exercise.

**ACTIVITY 1.5**

**Aim:** To print data generated by LM35 on serial monitor of NodeMCU and blink LED when temp gets change.

**Schematic diagram:**

****

**Program:**

int outputpin= A0;

void setup() {

Serial.begin(9600);

pinMode (D4, OUTPUT);

}

void loop()

{

int analogValue = analogRead(outputpin);

float millivolts = (analogValue/1024.0) \* 3300; //3300 is the voltage provided by NodeMCU

float celsius = millivolts/10;

Serial.print("in DegreeC= ");

Serial.println(celsius);

delay(1000);

if (celsius >= 35) {

digitalWrite(D4, HIGH); }

else {

digitalWrite (D4, LOW);}

}

**Outcome:**

|  |
| --- |
| A picture containing text  Description automatically generatedA picture containing text  Description automatically generated |
|  | Graphical user interface, text, application, email  Description automatically generated |

**Results:**

We successfully completed the activity and blinking the led when the temperature was high, and we obtained our findings on the board as well as the temperature shown on the serial monitor.