	RV College of Engineering® Department of Computer Science and Engineering CIE - II: Test and Quiz Paper		
Course & Code	IOT and Embedded Computing (CS344AI)		Semester: 4th Sem BE
Date : July 2024	Duration:120 minutes	Max.Marks:(10+50)=60 Marks	Staff : KB, SDV, MSS, MH
USN :	Name :		Section : A/B/C/D/CD/CY

NOTE: *Answer all the questions from Part-A (10 M) and Part-B (50 M)*

Sl.no	PART - A	Marks	* BT	*CO
1	Indicate the value to be loaded into match Register MR0, so that timer counter T0TC reaches the MR0 value after 5 milliseconds. Assume the PCLK = 10MHz, CCLK=40MHz, T0TC=0, Pre-scaler Register=0 Ans: 50000	2	L2	CO3

2	Calculate the delay produced by the following program run on LPC2148. Given PCLK = 15MHz. Choose the answer in milli-seconds.
	void delay(void)
	{
	T0MCR = 0X04;
	T0TC = 0X00;
	T0MR0 = 75000;
	T0TCR = 0X01;
	while(T0TC != T0MR0);
	T0TCR = 0X02;
	}
	
	Ans: 5ms
		2	L3	CO2

3	Given PCLK=15MHz, Required baud rate=9600, Choose the values of DLM:DLL. (Assume DivVal=0, MulVal=1) Ans: U0DLM=00;U0DLL=97;	2	L2	CO2
4	What are the different types of communication models used in IoT.	2	L2	CO2

	Ans:· Request-Response Communication Model · Publisher-Subscriber Communication Model · Push-Pull Communication Model · Exclusive-Pair Model			
5	List any four most commonly used sensors in IoT and mention any two applications of PWM in IoT Ans: Sensors- Temperature, Humidity, Moisture, Air Pollution, Vibration PWM Applications: LED Lighting, Servo Motor Control, DC Motor Control	2	L3	CO3

Sl.no.	PART - B	Marks	* BT	*CO
1a.	Generate the 200KHz, 25% duty cycle waveform using LPC 2148 PWM channel. Assume PCLK = 15MHz. Make suitable assumptions, and explain clearly the calculations and the working of the program. Assume PCLK = 15MHz T1 = Time Period of 200KHz = 1/20KHz = 0.005 msec T2 = Time Period of PCLK = 1/PCLK = 0.067 Microsecs No. of PCLKs required for one Timer period of 0.05ms= T1/T2 = 0.05msec/0.067Microsec= 74 to be loaded in MR0 register Assume PWM3 and PWM6 are used for generating waveforms with different duty cycle ratios, MR3 = 0.25 x 74 = 18 (25 % duty cycle) #include <LPC214x.h> void PWM_Init(void) #include <LPC214x.h> void PWM_Init(void)	5	L2	CO2

	{
	
	//P0.1 pin has second alternate function as PWM3 channel, so using PINSEL0 register
	
	PINSEL0 |= 0x00000008; // Select P0.1 as PWM output , bits D2 & D3 are for P0.1
	
	PINSEL0 |= 2 << 18; *//select P0.9 as PWM6 (option 2)*
	
	//Configure PWM channel 3 & 6 as single edge type and enable the channel
	
	PWMPCR = (1<<11) | (1 << 14);
	
	//load the value to MR0 to fix the pulse rate
	
	PWMMR0 = 74; // 200KHz pulse rate
	
	// enable PWM unit of LPC2148 and start the timer
	
	PWMTCR = 0x0000 0009; // bit D3 = 1 (enable PWM), bit D0=1 (start the timer)
	
	}
	
	int main()
	
	{
	
	PWM_Init();
	
	while(1)
	
	{
	
	PWMMR3 = 18; //25% duty cycle
	
	PWMLER = 0X48; // enable for channel 3 and 6
	
	}
	
	}			

1b.	Generate the 10KHz square waveform using LPC 2148 GPIO pin P0.1. Use timers to calculate the timings and assume PCLK = 60MHz. Explain the working of the program	5	L2	CO2

Td =   1/10KHz = 0.1 msec , half of it is 0.05msec;

T = 1 / PCLK = 1 / (60MHz) = 0.01666

micro seconds
count   =

Td / T = 0.05 msec/0.0166 micro = 3000

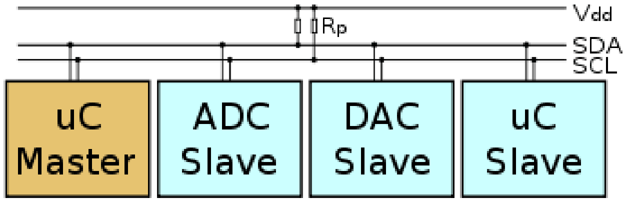
```
int main(void)

{

     T0MR0 = 3000 ;  //use the Timer0 and

load the MR0 with count

T0MCR = 0X0004;  //   0000….100 – Stop

the timer, after match

     I0DIR0 = 0X00000002; //make P0.1 as output

while(1)  // program to produce square

waveform  of 1 KHz

               {

               I0SET0

  =  1 << 1;  //set P0.1 to 1

               delay(

);

               I0CLR0

=  1 <<1;  //clear P0.1 to 0

delay(

);

               }
```

```
      }

      void   delay(void)

      {

        T0TCR = 1;  //start the timer

        While (!(T0TC == T0MR0));

  T0TCR = 2; // reset the counter  and stop the timer

       }
```

2a.	**Design an activity LED (one which is blinking once in 10 seconds to indicate the system/product is working) using interrupts and timers, with suitable comments**	5	L2	CO3

```
#include <LPC2148x.h>
unsigned int x=0;
__irq void Timer0_ISR(void) // an ISR program
{
x = x ^ 1;
if (x)
 I0SET1 = 1 << 16; //P1.16 = 1
else
 I0CLR1 = 1 <<16; // P1.16 = 0
T0IR = 0x01; // clear match0 interrupt, and get ready for the next
interrupt
VICVectAddr = 0x00000000 ; //End of interrupt
}
int main(void)
{
I0DIR1 = 0x0001 0000; //set P1.16 as output
T0TCR = 0x00; // stop the timer, to initialize different registers
T0MCR= 0x0003; // Enable Interrupt and reset timer after match
T0TC = 0x00; // make TC = 0
 T0MR0 = 150000; // generates 10ms
//load interrupt related registers , assigning Timer0 to IRQ slot 4
 VICVectAdd4 = (unsigned long)Timer0_ISR; // set the timer ISR vector
address
VICVectCntl4 = 0x0000024; // set the channel
VICIntEnable = 0x0000010; // enable the timer0 interrupt

T0TCR = 0x01; // start the timer
 while(1)
 {
 //do other works
```

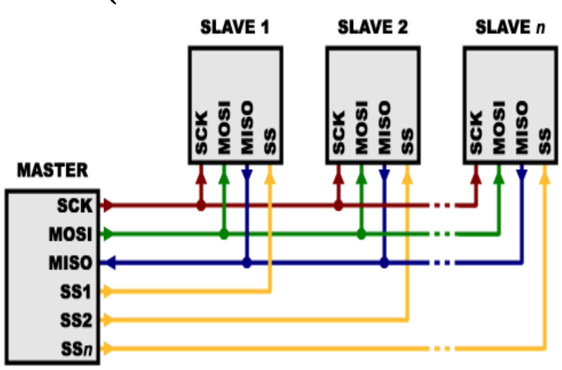
<table>
<tr><td></td><td>}; // now timer interrupt is serviced automatically using the ISR
}</td><td></td><td></td><td></td></tr>
<tr><td>2b.</td><td>Discuss the Features and Applications of serial protocols I2C and SPI



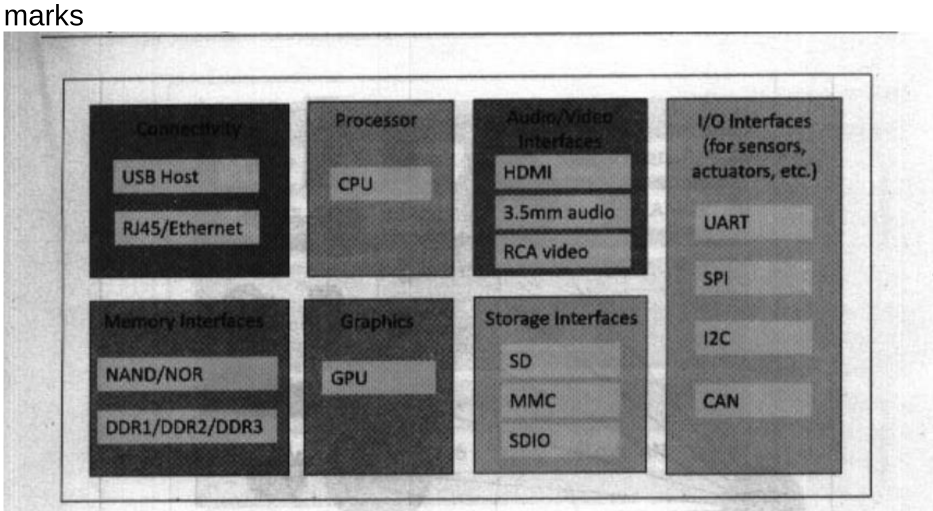
I2C Features

- It is multi-master, multi-slave, packet switched, single-ended, serial computer bus. It is widely used for attaching lower-speed peripheral IC's to processors and microcontrollers in short-distance, intra board communication.
- Support multi master system, more than one master can communicate with the devices, for every 8 bits of data sent, one extra bit of meta data (ACK/NACK bit) must be transmitted. ACK/NACK bit gives confirmation that each frame is transferred successfully.
- Only uses two wires (Serial, half duplex), Hardware is less complicated than with UARTs. The hardware required to implement I2C is more complex than SPI but simpler than UART.
- It Supports 128 devices (7bit address) in normal mode.
- Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode. (It supports 400Khz (fast mode), 1Mhz –fast mode plus,3.4Mhz for high speed mode, 5Mhz for ultra fast mode)

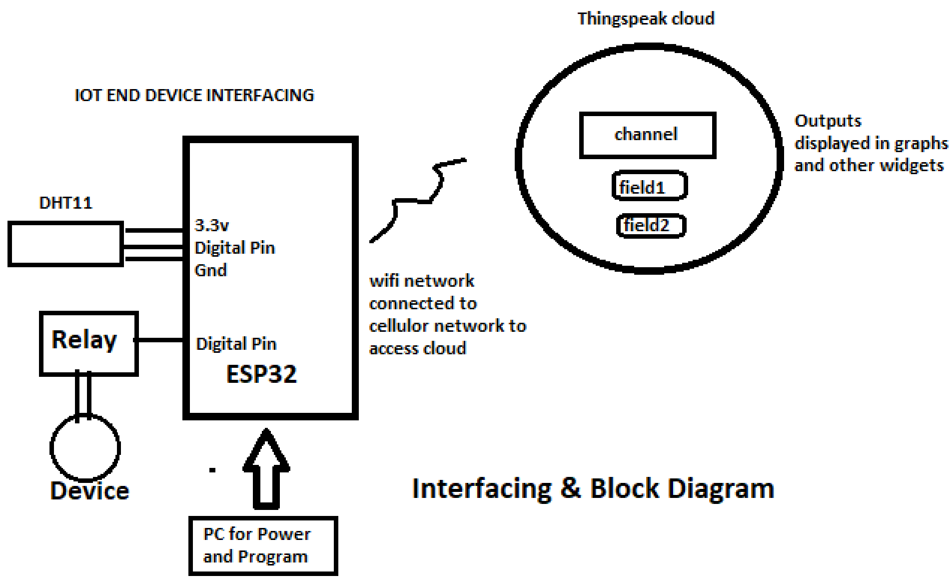
SPI (Note: connections for one device good enough)



- SPI Interface uses four wires for communication. Hence it is also known as four wire serial communication protocol.</td><td>5</td><td>L3</td><td>CO1</td></tr>
</table>

	➢ SPI is a full duplex master-slave communication protocol. This means that only a single master and a single slave can communicate on the interface bus at the same time. It has separate send & receive lines unlike I2C. ➢ SPI enabled devices work in two basic modes of SPI operation i.e. SPI Master Mode and SPI Slave Mode. Master Device is responsible for initiation of communication. Master Device generates Serial Clock for synchronous data transfer. There is always only one master (most of the times it is microcontroller). ➢ Faster than asynchronous serial (UART), operate around 1Mhz. (can go upto 10Mhz) ➢ Hardware requirement for SPI is very simple (as simple as shift register) compare to UART & I2C. ➢ Master Device can handle multiple slave devices on the bus by selecting them one by one using multiple slave select pins. In general, each slave will need a separate SS line.			
3a.	Define IoT and Explain the functional blocks of IoT with the help of a neat block diagram. Definition of IoT - 1 Mark, Block diagram - 2 Marks, Brief explanation-2 marks  Figure 1.3: Generic block diagram of an IoT Device	5	L3	CO3
3b.	Suggest (With brief description) any one-use case of IOT pertaining to following domains: Energy, Retail, Logistics, Agriculture, Cities. Any one-use case: Block diagram representation (optional) + Explanation	5	L4	CO1

<table>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>4a.</td><td>
Design an IOT Level 2 deployment application for weather monitoring and Device control in the house using ESP32 and Thing speak cloud platform, with suitable block diagram, interfacing, flowcharts and brief description. The proposed system consists of single node that monitors the room temperature and humidity using DHT 11 sensor, and based on the temperature / humidity, device(fan) should be turned on using a Relay. The controller also sends the sensor data to the cloud, where it will be displayed on the dash board.



Interfacing & Block Diagram

Algorithm:

On the Cloud:

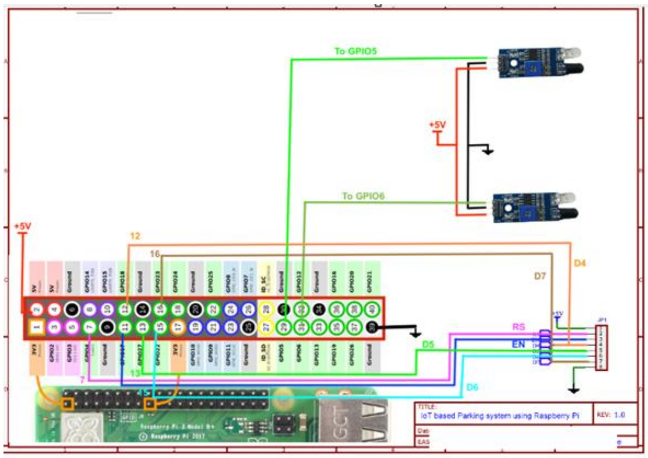
1.Goto thingspeak cloud, do following things: Create the channel and two fields for storing temperature and humidity

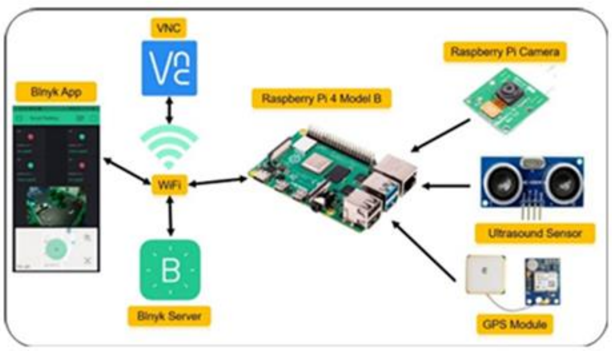
2.Copy the Channel no and write API to be used at the IOT Edge device.

On the End Device:

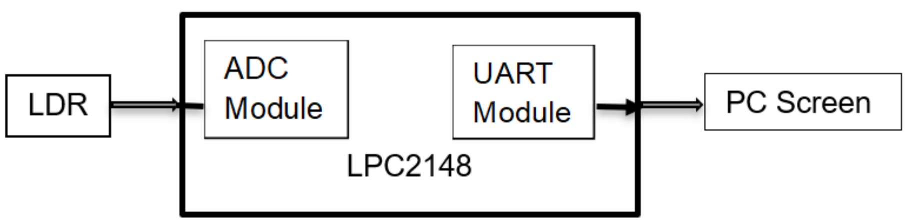
1. Import the Libraries for DHT11,ThingSpeak cloud
2. Initialize the required variables, libraries (start functions) with appropriate digital pins used for interfacing. Set the temperature threshold for fan/device control
3. Read the temperature and humidity
4. Upload the values to the thingspeak cloud channel fields, using the channel ID and fields.
5. Compare the room temperature with threshold and switch on / off the relay to control the device/fan.
6. delay, after each reading
</td><td>10</td><td>L4</td><td>CO2</td></tr>
</table>

	7. repeat the steps 3 - 6			
4b.	Design an IOT Leve2 deployment application for Smart Parking using RasberryPie with IR sensors and Cloud with Mobile Application to show the parking slots status. Draw the block diagram, interfacing, flowchart and brief description. **Firebase Configuration:** Configuring Firebase involved several steps: ● In the Project settings under the Firebase Admin SDK section, we generated a new private key and saved the key.json file to our project directory. This key was essential for authenticating and interacting with Firebase services. Firebase was used to store real-time data on parking spot availability, manage user authentication, and log vehicle entry and exit times. Component Setup: ● Raspberry Pi 4: Served as the central hub for managing the parking system, running the application, and interfacing with hardware components. ● Ultrasonic Sensors: Installed at each parking spot to detect the presence of vehicles. These sensors were connected to the Raspberry Pi GPIO pins. ● Camera Module: Used to capture images of vehicles entering and exiting the parking area for license plate recognition. ● LED Indicators: Installed to show the status of each parking spot (occupied or available). ● Display Screen: Provided real-time information on parking availability to users at the entry point. Hardware Configuration: Raspberry Pi GPIOs are connected to the ultrasonic sensors and LED indicators. The pinout diagram was essential to ensure the connections were correct and avoid any potential hardware damage. The camera module is secured and positioned to capture clear images of vehicle license plates.	5	L4	CO2

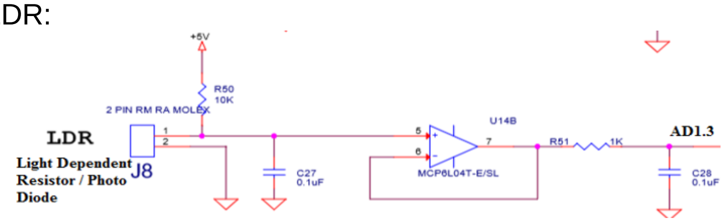




5 Interface LDR and LED bulb to LPC 2148 and write an embedded C program to read the data from LDR and suitably turn on/off the LED bulb and also send the suitable message to computer using UART interface. Clearly show the connections between LPC 2148 and Computer Serial Port and explain the UART initialization steps, clearly showing the registers used and the baud rate calculations.



LDR:




```
AD0CR=0x00200600|(1<<ch);      //select channel
```

	AD0CR|=(1<<24); //start conversion
	while((AD0GDR& (1U<<31))==0);
	val=AD0GDR;
	
	U0THR =val; //send to serial port(check on the terminal)
	
	UART Initialization::
	void uart_init(void)
	{
	//configurations to use serial port
	PINSEL0 |= 0x00000005; // P0.0 & P0.1 ARE CONFIGURED AS TXD0 & RXD0
	U0LCR = 0x83; /* 8 bits, no Parity, 1 Stop bit */
	U0DLM = 0; U0DLL = 8; // 115200 baud rate
	U0LCR = 0x03; /* DLAB = 0 */
	U0FCR = 0x07; /* Enable and reset TX and RX FIFO. */
	}			

Course Outcomes: After completing the course, the students will be able to:-	
CO 1	Apply Embedded System and IoT fundamentals and formulate sustainable societal relevant cost effective solutions.
CO 2	Demonstrate the development of software programs using Embedded C, using Microcontrollers and different sensors and peripherals to build embedded system applications.
CO3	Design smart systems using various I/O peripherals, Sensors, embedded protocols like UART,I2C,SPI using modern tools like Keil IDE software for various domains like Healthcare, automation, agriculture, smart cities and others.
CO 4	Indulge in developing Novel multi-disciplinary IoT projects using prototype boards, with effective oral & written communication skills and working in teams.
CO 5	Engage in Lifelong Learning by investigating and executing real world societal problems using engineering tools – Cross compilers, debuggers and simulators, emerging processor and controller-based hardware platforms, IOT cloud infrastructure & protocols.

BT LEVELS	L1	L2	L3	L4	L5	L6	COS	CO1	CO2	CO3	CO4
MARKS		10	30	10					20	30	