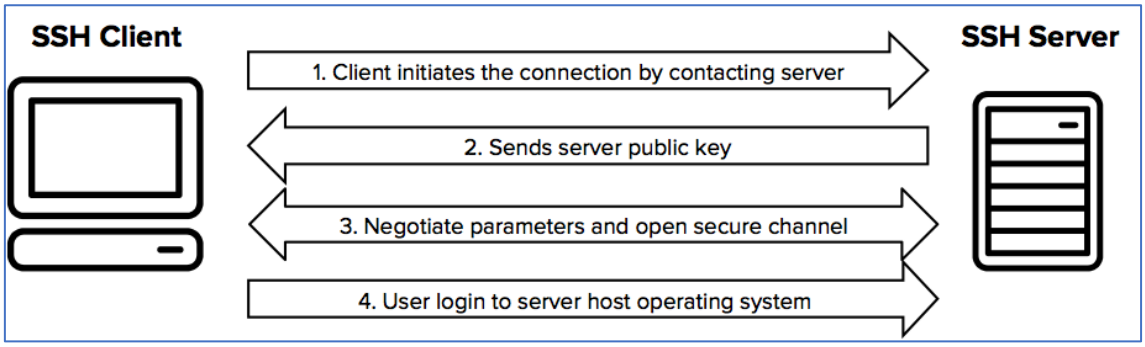
LABORATORY 4: SSH and Timers

Part 1: SSH.

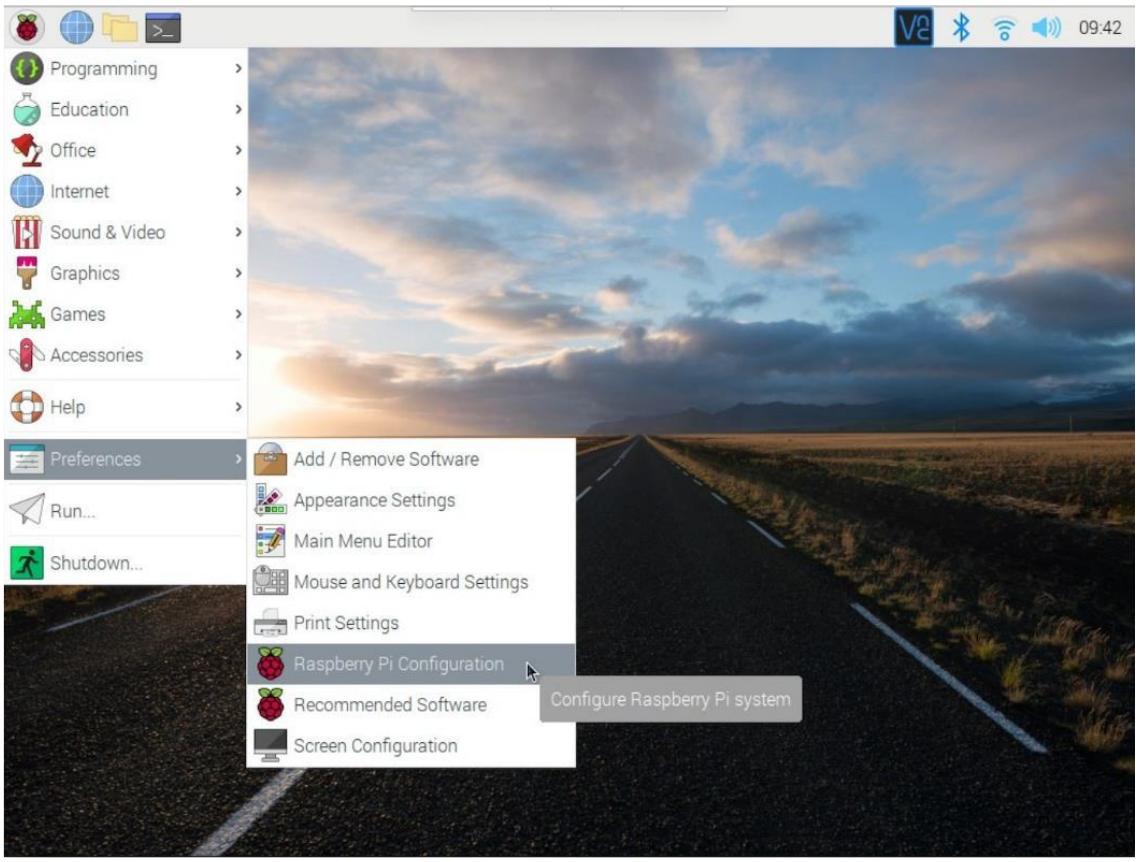
Theory Concepts:

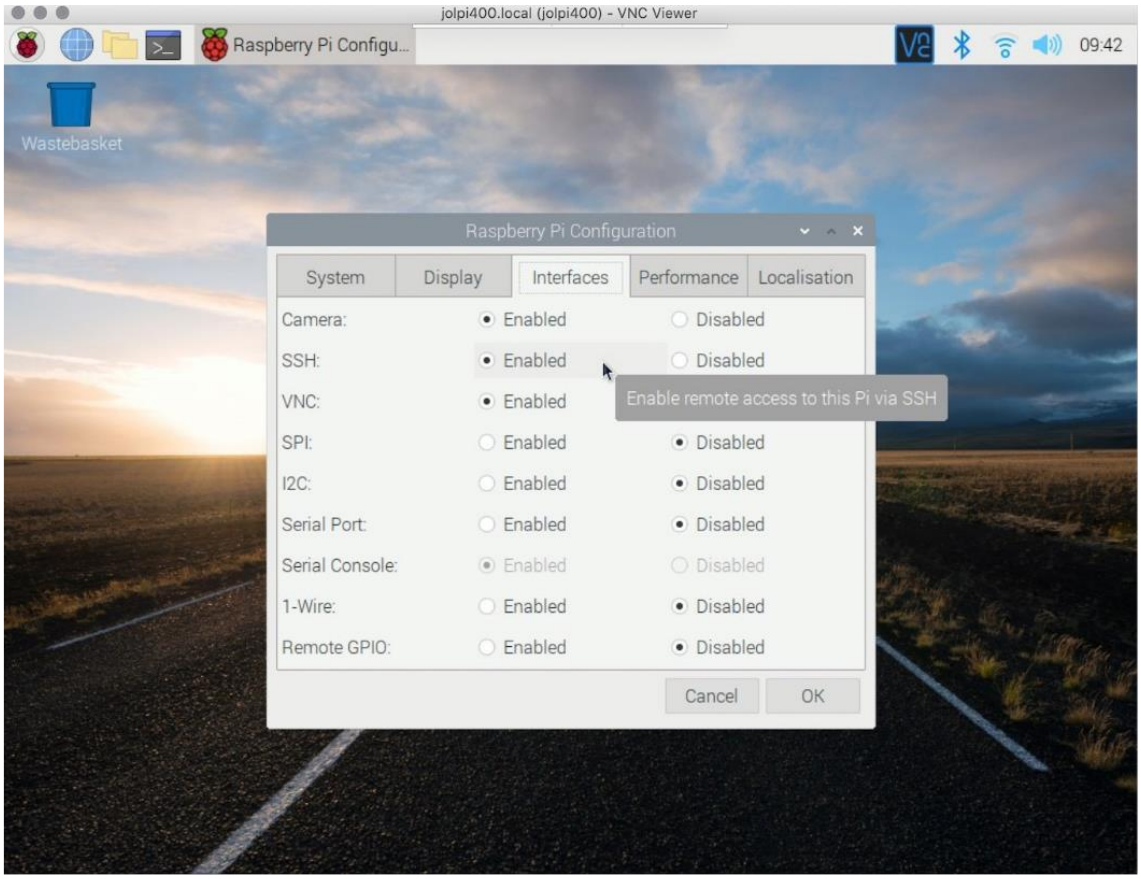
SSH comes from “Secure Shell” and is a remote Administration protocol that enables the control of a remote device.



Ssh needs a hostname/ip, port, user credential and password.

We need to enable ssh on our raspberry Pi:





After enabling ssh on the raspberry pi we can access to this device remotely using the ip address it has and the user password. To do this a common network between the raspberry and the computer is needed.

We can follow the next steps to connect and disconnect using ssh.

1. Enable Raspberry PI's SSH from Raspberry Pi configuration -> Interfaces -> Enable SSH
2. It's recommended to change the Raspberry password using the command **sudo raspi-config.**
3. Raspberry PI's IP can be showed using the command **hostname -I**.
4. Use the command **ssh pi@[raspberry_IP]** in Ubuntu like for example: ssh pi@111.111.1.111.
5. Usually after step 4 is needed to put the Raspberry password and username.
6. To end SSH communication, in the terminal type "exit".

Which is the default port ssh uses?

Can we connect to our raspberry pi using different devices than our computers like our smartphones? If so, how can we do it?

Exercises:

- Using SSH create a new folder in your raspberry pi with the name "lab4", then make a python file inside the folder with the name "buzzer.py"

- Generate a python program in your computer that makes a buzzer sound when a button is pressed.
- Using SSH modify the previous program to make the buzzer sound when a button is pressed and stop when a second button is pressed.
- Create a led sequence that has a variable time interval between states. Make the program in a way that it can get ssh modifications without stopping the execution. Tip: Modify a txt file with ssh and read the file values in the main program from Raspberry Pi.

Can you program in python using SSH from your computer?

Investigate how VSCode can manage to work remotely using the 'Remote-SSH extension'.

Part 2: Timers.

- Timers are used to generate operations according to some events during the time.
- Generally, we will generate interruptions to the main routine when a timer is executed.
- We can use delay to interrupt all the program execution and timers to change the main routine for a sub routine.

```
uint32_t FS= 120000000*2;
void timer0A_handler(void);
uint8_t switch_state = 0;
uint32_t button = 0;
int main(void)
{
    //Clock Configuration
    SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480),120000000); // Eables system clock
    //Enable peripherals
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION); //Enables peripheral N
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //Enables peripheral F
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    //Enable Timer Peripheral
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
```

We must set the timer to work correctly. For that, is needed to define a Frequence Sample or FS that works as a Prescaler for our timer. The following formula can help us to select the proper FS:

$$timerFreq = clockfreq / FS$$

With the FS selected we have to start the processor interruption and then enable the timer:

```
//Set timer
TimerConfigure(TIMER0_BASE,TIMER_CFG_PERIODIC);
//Set the cout time for Timer
TimerLoadSet(TIMER0_BASE, TIMER_A, FS);
//Enable processor interrupts
IntMasterEnable();
//Enable Interrupt
IntEnable(INT_TIMER0A);
//Enable timer A interrupt
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
//Enable the timer
TimerEnable(TIMER0_BASE,TIMER_A);
```

The timer uses an external function to execute the sub routine. This routine will stop the main routine and depending on if the timer is periodic, or one shot will keep executing.

```
void timer0A_handler(void)
{
    switch_state ++;
    //Changing the timer delay
    if (GPIOPinRead(GPIO_PORTJ_BASE, GPIO_PIN_0) == 0)
     {
      button ++;
     }
    if(button >0)
    {
          FS =120000000*4;
          TimerLoadSet(TIMER0_BASE, TIMER_A, FS);
    }
    //Clear timer
    TimerIntClear(TIMER0_BASE, TIMER_A);
    // Sequence 1.
    if(switch_state % 2 != 0)
    {
         GPIOPinWrite(GPIO_PORTN_BASE,0x03,0x02);  //set periph base,pins, values
         GPIOPinWrite(GPIO_PORTF_BASE,0x11,0x01);  //set periph base,pins, values
    }
    // Sequence 2.
    if(switch_state % 2 == 0)
    {
          GPIOPinWrite(GPIO_PORTN_BASE,0x03,0x01);
```

We must declare the timer function in the startup file of the project.

```
//*****************************************************************************
//
// The entry point for the application.
//
//*****************************************************************************
extern int main(void);
extern void timer0A_handler(void);
//*****************************************************************************
```

```
3     IntDefaultHandler,                      // ADC Sequence 0
4     IntDefaultHandler,                      // ADC Sequence 1
5     IntDefaultHandler,                      // ADC Sequence 2
6     IntDefaultHandler,                      // ADC Sequence 3
7     IntDefaultHandler,                      // Watchdog timer
8     timer0A_handler,                      // Timer 0 subtimer A
9     IntDefaultHandler,                      // Timer 0 subtimer B
0     IntDefaultHandler,                      // Timer 1 subtimer A
1     IntDefaultHandler,                      // Timer 1 subtimer B
```

This process will help us to generate a timer sequence and use it to make an interruption to the main process.

Exercises:

- Blink the user LED PN1 with **timer interruptions** in the Tiva. Initially blink the LED every second, then modify the code to blink the LED every 2 seconds and finally change the code to blink the LED every 5 seconds.
- Make a decimal sequence with the user leds that changes each 2 seconds.
- Modify the previous excercise to change the interval to 1 and 0,5 seconds when a user switch is pressed.
- Modify the binary counter exercise from lab 3 to use timers, each sequence should change without buttons and in a 1.5 second interval. Then add a new functionality, when a button is pressed change the time interval to 3 seconds.