**Практическая работа №1**

**Основы работы с микрокомпьютерами серии PI**

**Цель работы:**

Выполнение практической работы направлено на изучение:

1. принципов предварительной настройки микрокопьютера;

2. принципов подключения, сбора и обработки данных с различного периферийного оборудования;

**Порядок работы:**

1. Установили с использованием ПО *Rufus* образ ОС *ArmBian* на флеш-карту, подключили ее к микро-ПК, подключили к нему дополнительно *HDMI*-монитор, *USB*-клавиатуру, *USB*-мышь и блок питания. Задали пароль для *root* пользователя, имя и пароль основного пользователя.

Запустили  *user-friendly* псевдографический менеджер настройки ОС

**sudo armbian-config**

Затем настроили SSH доступ (*Secure* *SHell* — сетевой протокол, позволяющий соединяться с удалённым сервером и выполнять на нём команды, загружать файлы) и подключение микро-ПК по *WiFi* к сети интернет.

2. Выполнили базовые обновления системы, установку дополнительных компонентов

**sudo apt-get update**

**sudo apt-get upgrade**

Важным вопросом является достаточность ОЗУ для решения всех планируемых задач. В плате *Orange PI Lite* установлено только 512 Мб ОЗУ (фактически 492 Мб). Расширением ОЗУ в ОС является файл подкачки (в *Windows*) или swap в ОС *Linux*. После установки ОС в *Orange PI Lite*, размер swap составляет по умолчанию 246 Мб. Возможным путем расширения является использование в качестве swap внешнего *USB*-накопителя.

Проверили диск на остаток свободного места для его определения как ***swap***

**sudo cfdisk /dev/mmcblk0**

Последняя часть команды – указание физического диска (флеш-карты).

После создания на свободном месте диска дополнительного раздела (***/dev/mmcblk0p2***) обозначили его как swap. Для этого ввели команду

**sudo blkid**

Размонтировали раздел от системы и сделали его ***swap*** командами

**sudo umount /dev/mmcblk0p2**

**sudo mkswap /dev/mmcblk0p2**

Запустили текстовый редактор на файле ***/etc/fstab***

**sudo nano /etc/fstab**

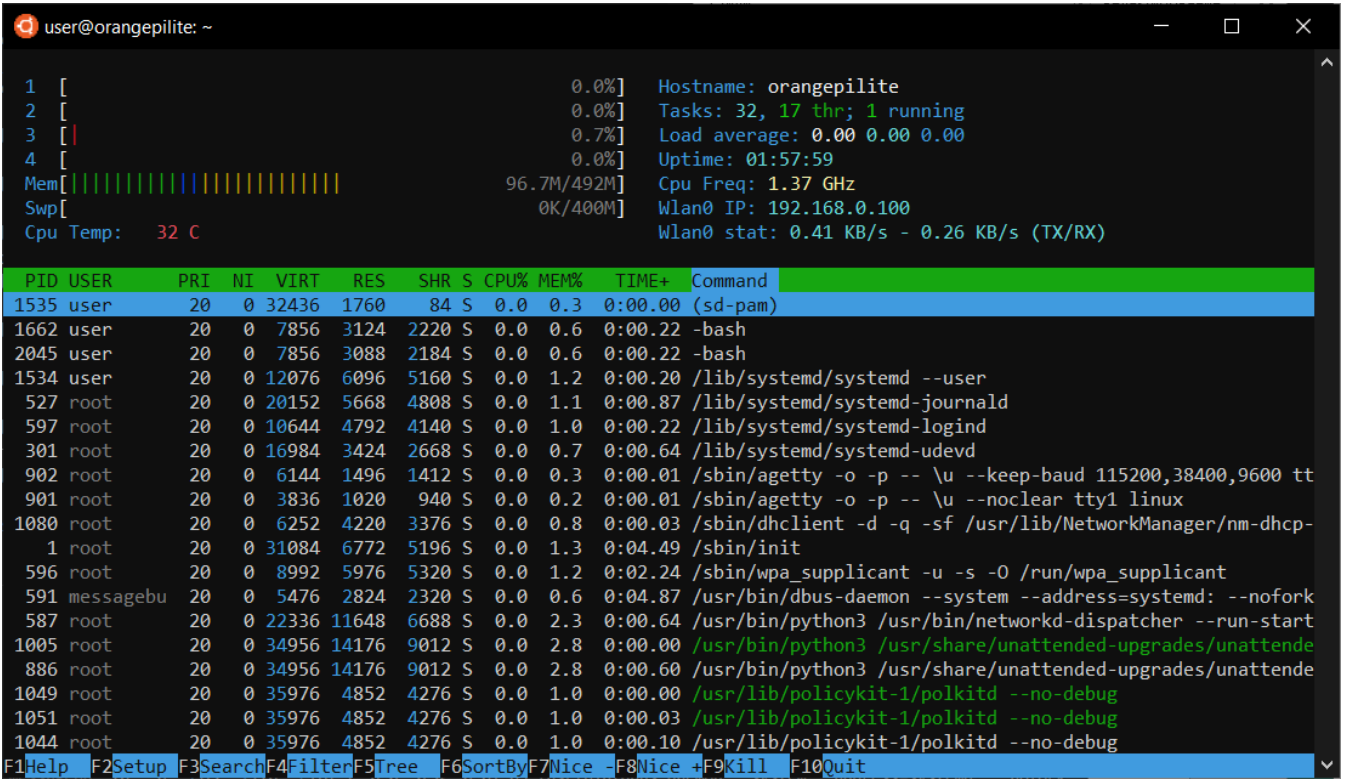
Выполнили обновление информацию

**sudo swapon -a**

Проверили размер ***swap*** программой ***htop***, которую установили командой

**sudo apt-get install htop**

как показано на рисунке 1.

Рисунок 1 - Экран программы ***htop***

***Swp*** составляет 400*Mb*. Этого объема должно быть достаточно.

3. Для работы с видеокамерой и установки ***tensorflow*** выполнили установку необходимых библиотек с помощью команд

**sudo apt-get install python3-dev python3-pip libhdf5-dev libc-ares-dev libeigen3-dev libatlas-basedev libopenblas-dev libblas-dev liblapack-dev cython3**

**sudo apt-get install default-jdk automake autoconf**

**sudo apt-get install curl zip unzip libtool swig libpng-dev zlib1g-dev pkg-config git g++ wget xz-utils**

**sudo apt-get install python3-numpy python3-dev python3-pip python3-mock**

**pip3 install -U --user keras\_applications==1.0.8 --no-deps**

**pip3 install -U --user keras\_preprocessing==1.1.0 --no-deps**

**pip3 install portpicker**

**sudo apt-get install libpython3-all-dev:armhf**

**sudo apt-get install python3-opencv protobuf-compiler python3-pygame**

**sudo pip3 install opencv-python**

Подключили физически видеокамеру в порт. Создали программу, осуществляющую снимок с видеокамеры и сохраняющую изображение под именем «*filename.jpg*»

**Import pygame**

**import pygame.camera**

**pygame.camera.init()**

**camlist = pygame.camera.list\_cameras()**

**cam = pygame.camera.Camera(camlist[0], (640, 480))**

**cam.start()**

**image = cam.get\_image()**

**pygame.image.save(image, “filename.jpg”)**

**cam.stop()**

4. Так как ОС микро-ПК имеет текстовый интерфейс и просмотреть содержимое файла в привычном виде не представляется возможным, осуществили вывод изображения с микрокомпьютера посредством *Telegram*-бота. Для этого создаем программу *cam\_look\_bot.py.*

**pip3 install telepot**

**import telepot //Модуль для работы с Telegram Bot API**

**import time //Модуль для обработки задержек**

**import pygame //набор модулей для написания мультимедиа-приложений**

**import pygame.camera //модуль для работы с видеокамерой**

**def handle(msg): //функция для обработки входящих сообщений**

**chat\_id = msg['chat']['id']**

**command = msg['text']**

**print('Got command: %s' % command)**

**print('From : %s' % chat\_id)**

**if command == '/command1':**

**bot.sendMessage(chat\_id, 'Oks')**

**elif command == '/command2':**

**bot.sendMessage(chat\_id, 'Ok')**

**elif (chat\_id== 1263505874 ) & (command == '/photo'):**

**cam.start()**

**image = cam.get\_image()**

**cam.stop()**

**pygame.image.save(image, "filename.jpg")**

**bot.sendPhoto(chat\_id, photo=open('filename.jpg', 'rb'))**

**pygame.camera.init()**

**camlist = pygame.camera.list\_cameras()**

**cam = pygame.camera.Camera(camlist[1], (640, 480))**

**bot = telepot.Bot('6631506081:AAEIe\_Y9corp0If2WQ7ZPY\_PF9BjUHCLknk')**

**bot.message\_loop(handle)**

**print('I am listening ...')**

**while 1:**

**time.sleep(10)**

Функция *handle* используется в качестве обработчика входящих сообщений в Telegram. Она извлекает *chat\_id* и *command* из сообщения. Если command соответствует "/*command1*", отправляется сообщение "*Oks*" на *chat\_id*. Если соответствует "/*command2*", отправляется сообщение "*Ok*". Далее выполняется проверка, является ли *command* "/*photo*", а *chat\_id* соответствует нашему идентификатору *Telegram*. Если так, то программа запускает камеру, захватывает изображение, сохраняет его в файле "*filename.jpg*" и отправляет фото в чат с использованием *bot.sendPhoto.*

Далее происходит инициализация модуля камеры pygame, получение списка доступных камер и создание объекта камеры с использованием второй камеры в списке. Размер изображения камеры установлен на 640x480 пикселей.

Затем создается объект *bot* с помощью нашего токена *Telegram Bot*. Запускается цикл, непрерывно слушающий входящие сообщения для передачи их в функцию *handle*.

Программа входит в бесконечный цикл для обеспечения постоянной работы бота и непрерывного прослушивания входящих сообщений от пользователей. Внутри цикла присутствует задержка, которая приостанавливает выполнение программы на 10 секунд перед следующей итерацией цикла. Это позволяет снизить нагрузку на процессор и сеть, а также предотвратить слишком частые проверки на наличие новых сообщений.

**Вывод:** изучили принципы предварительной настройки микрокомпьютера *Orange PI Lite*, принципы подключения, сбора и обработки данных с внешней видеокамеры, получили навыки работы с библиотеками *tensorflow*, *OpenCV*, *pygame*, *telepot*, научились осуществлять удаленное подключение к плате микрокомпьютера по протоколу *SSH*, получили навыки создания *Telegram*-ботов и осуществления с их помощью сбора информации с микрокомпьютера.