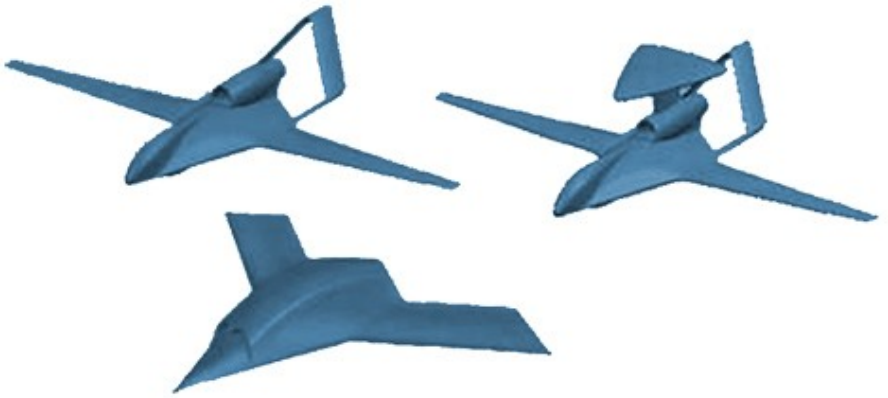
Лабораторная работа №11

Беспилотный летательный аппарат: Приложение для видеонаблюдения

Разработка мультиагентной системы для БПЛА

Беспилотные летательные аппараты



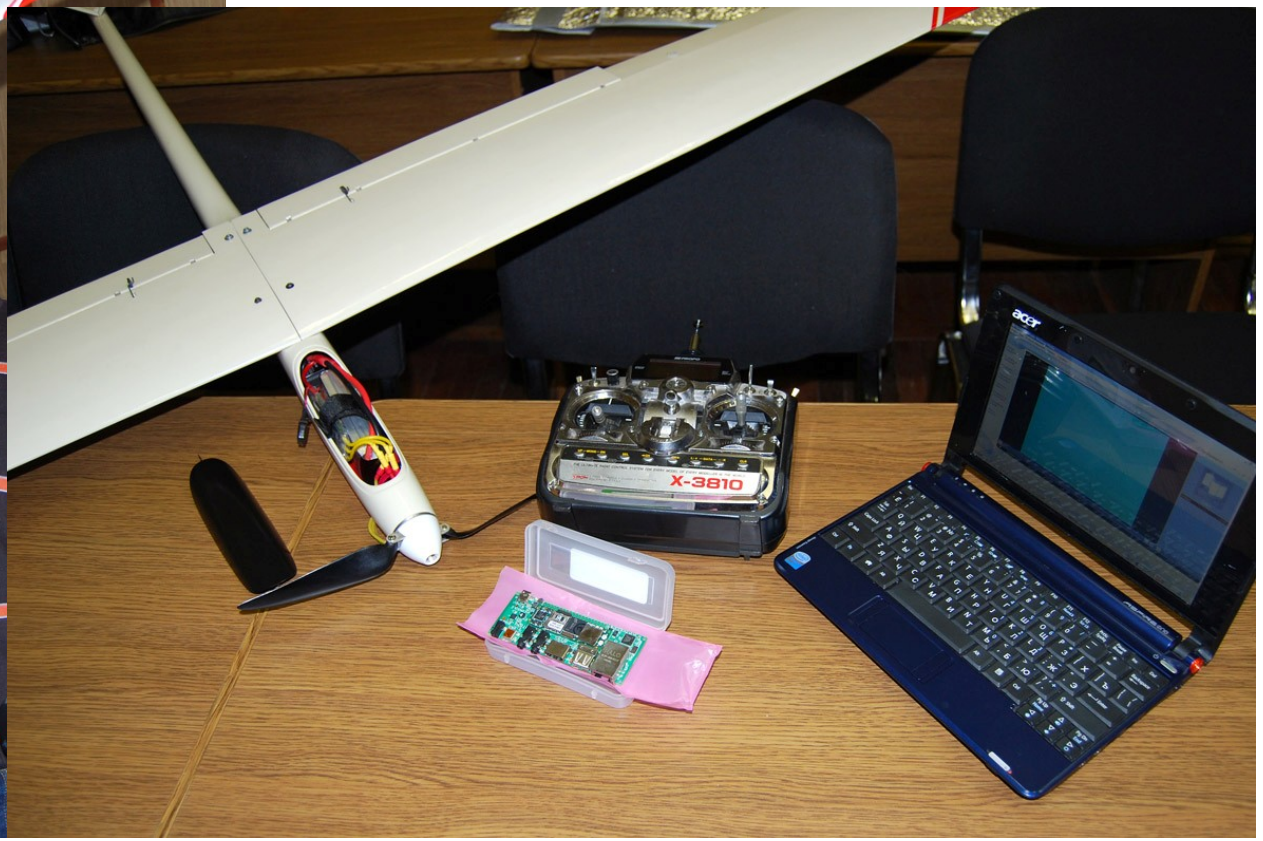
- Взаимодействие БПЛА с мобильной базовой станцией и между собой
- Бортовой микрокомпьютер
- Автопилот Paparazzi
- Архитектура мультиагентной системы

Мобильная базовая станция

Планер “PAPRIKA”

- Длина, мм: 1200
- Размах крыла, мм: 2006
- Площадь крыла, 35,8 кв. дм
- Максимальный полетный вес, г: 2000-2100;
- Полезная нагрузка, г: 600
- Крейсерская скорость, км/ч: 50
- Дальность полетов, км: 200

Внешний вид БПЛА и система управления



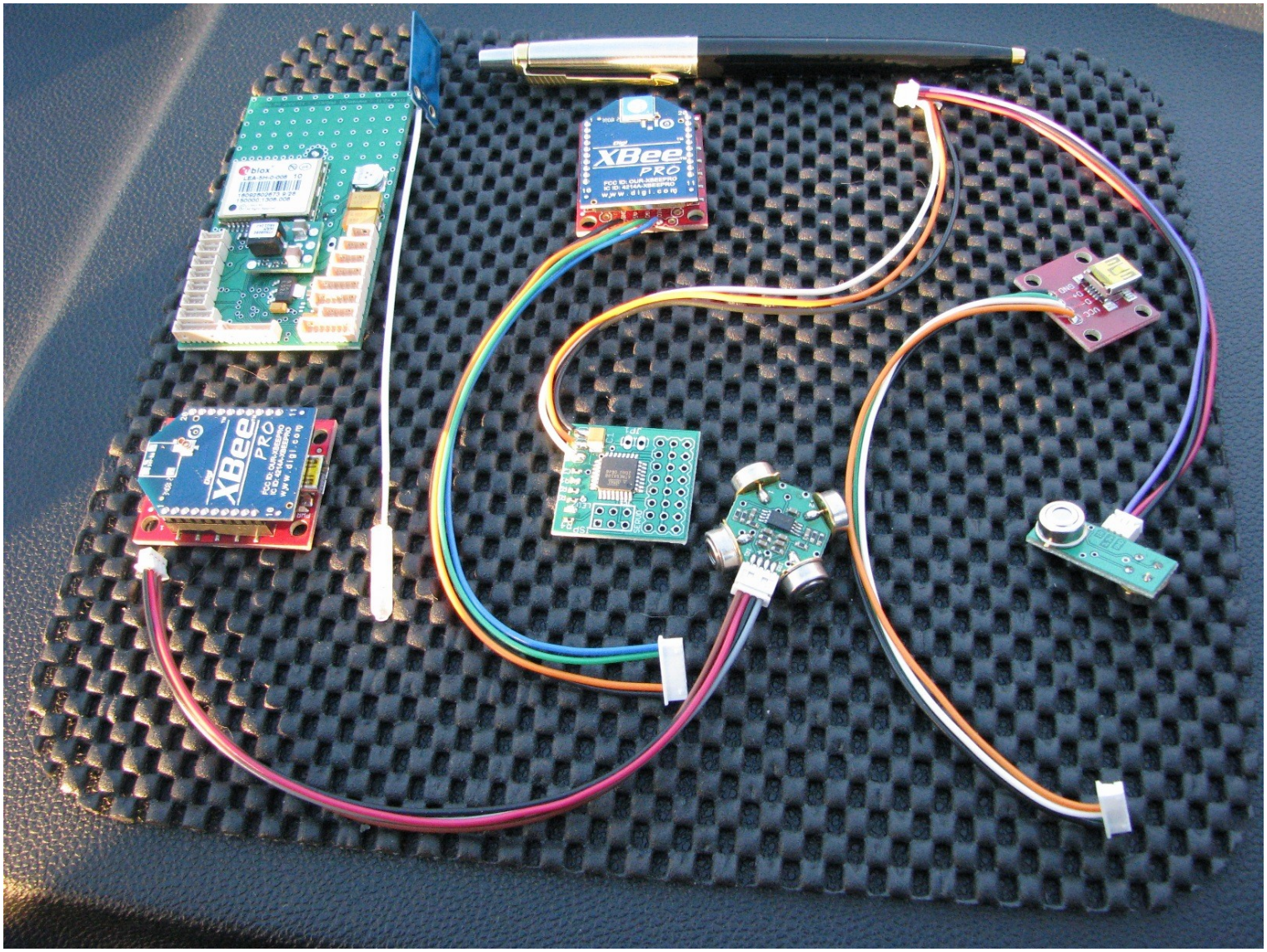
Бортовой микрокомпьютер

- Gumstix Overo Air
- Размер: 17 мм x 58 мм x 4,2 мм
- Процессор ARM Cortex-A8 600 Mhz
- 256 MB памятью RAM
- Встроенная память 256 MB NAND Flash
- слот для карты micro SD
- 27-pin слот для микро видеокамеры
- ОС Linux
- Связь 2,4 GHz с 802.11 a/b/n (wi-fi).
- Связь GPRS по GSM модему

Система навигации ArduIMU
Автопилот Papapazzi

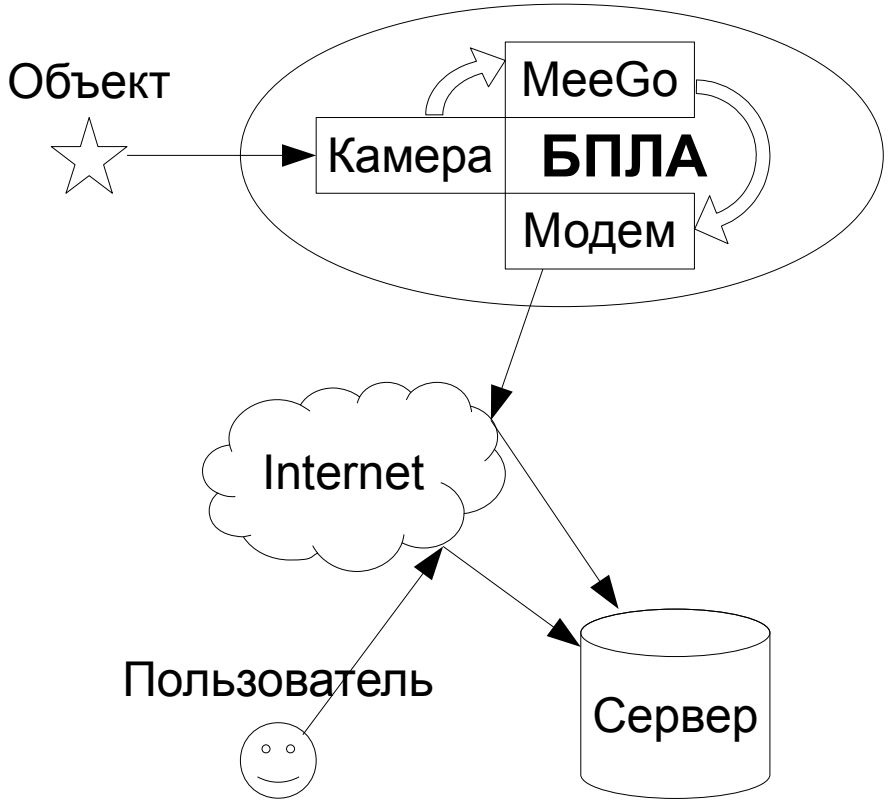
- микроконтроллер
- триада инерциальных датчиков
  (пъезогироскопы по трем осям)
- трехосный магнитометр
  (определение азимута движения)
- трубка Прандтля (скорость и высота)
- GPS модуль U-Box LEA-5E с частотой 4 Hz
- датчики горизонта
- автопилот
- дешифратор ШИМ

Автопилот Paparazzi



Приложение – постановка задачи

- Получить кадр от видеокамеры
- Отправить кадр на сервер через GSM USB-модем
- Сохранить кадр на сервере и обеспечить к нему доступ пользователей



Требования и окружение

- ОС БПЛА — MeeGo
- Разработка для MeeGo средствами самой MeeGo
- ОС сервера — Fedora/RHEL/CentOs/MeeGo, в других случаях используйте подходящий менеджер пакетов
- Камера — любая, с которой работает ядро MeeGo
- Модем — любой, мы использовали ZTE MF100 Beeline

Разбиваем задачу на подзадачи

1. Настроить подключение к Интернет через USB-модем
2. Сохранить кадр камеры в jpg-файл
3. Отправить jpg-файл на сервер
4. Принять jpg-файл на сервере и предоставить доступ пользователям

Настройка USB-модема – варианты

- **Непосредственная настройка демона pppd**
- Использование программ дозвона, например, wvdial
- Использование oFono

Настройка PPPD для USB-модема 1

Устанавливаем PPPD в MeeGo:

$> yum install ppp

Конфигурация pppd '/etc/ppp/peers/modem':

/dev/ttyUSB2 *460800*

connect '/usr/sbin/chat -v -f /etc/ppp/chat-modem'

defaultroute

replacedefaultroute

noauth

init 'echo -e "nameserver 8.8.8.8" > /etc/resolv.conf'

Настройка PPPD для USB-модема 2

Конфигурация chat '/etc/ppp/chat-modem':

TIMEOUT 5

ECHO ON

ABORT '\nERROR\r'

ABORT '\nNO CARRIER\r'

'' \rAT

TIMEOUT 12

*OK 'AT+CGDCONT=1,"IP","**your-providers-apn.net**"'*

*OK ATDT***99***1#***

CONNECT ''

Использование USB-модема

- Подключение к Интернет из консоли

  $> *pppd call modem*
- Отключение

  $> *kill `cat /var/run/ppp0.pid`*

Захват кадра с камеры – варианты

- **Использование интерфейса v4l2, поддерживаемого ядром MeeGo**
- Библиотека gStreamer
- Библиотека QT Phonon поверх gStreamer, штатно отсутствует в MeeGo
- QT Mobility/Multimedia, в будущем должна заменить Phonon

Сборка и использование v4l2grab

- Уже есть готовая программа!

  http://www.twam.info/linux/v4l2grab-grabbing-jpegs-from-v4l2-devices
- Устанавливаем зависимости

  $> *yum install gcc libjpeg-devel*
- Собираем прямо в MeeGo

  $> *gcc v4l2grab.c -o v4l2grab -ljpeg*
- Подключаем камеру и проверяем

  $> *./v4l2grab -d /dev/video0 -o image.jpg*

Отправка файла на сервер

- **Использование консольных HTTP-клиентов**, таких как **cUrl**, wget, …
- Использование библиотек HTTP-клиентов, таких как libcUrl.
- Использование QT Network

- **Стандартная связка Apache+PHP**
- **Простейший PHP-скрипт для приема файлов**

Настройка HTTP-сервера

- Устанавливаем и настраиваем Apache (www.apache.org) и PHP для него (www.php.net)

  $> *yum install httpd php*

- Даем скриптам права на запись в каталоге сервера

  $> *chmod a+w /var/www/html*

Скрипт для загрузки upload.php

```
<?php
if (@$_REQUEST['submitname']=="OK") {
    $target_path = "./".basename($_FILES['filename']['name']);
    if(move_uploaded_file($_FILES['filename']['tmp_name'], $target_path)) {
        echo "The file ".basename($_FILES['filename']['name'])." has been uploaded\n";
    }
    exit;
}
?>
<form enctype="multipart/form-data" action="upload.php" method="POST">
Choose a file to upload: <input name="filename" type="file" />
<input type="submit" name=submitname value="OK" />
</form>
```

Параметы HTTP клиента

- Устанавливаем cUrl в MeeGo

  $> *yum install curl*

- Отправляем image.jpg на server.com

  $> *curl -F submitname=OK -F filename=@image.jpg http://server.com/upload.php*

- Проверяем в браузере http://server.com/image.jpg

Итоговое приложение – варианты

- Скрипт, управляющий внешними утилитами
- Обычное C-приложение
- **QT-приложение с графическим интерфейсом пользователя**

SPRINT SPbSU-Intel Laboratory

Итоговое QT-приложение: camserver.h

Описываем новый Виджет с двумя слотами:

```
class CamServerWidget : public QWidget {
    Q_OBJECT
    public slots:
        void grab();
        void start();
    …
};
```

Итоговое QT-приложение: camserver.cpp – GUI

GUI задаем прямо в конструкторе:

```
CamServerWidget::CamServerWidget(QWidget *parent):QWidget(parent) {
    QPushButton *start = new QPushButton(tr("Start"));
    QPushButton *stop = new QPushButton(tr("Stop"));
    QPushButton *quit = new QPushButton(tr("Quit"));
    QVBoxLayout *layout = new QVBoxLayout;
    layout->addWidget(start);
    layout->addWidget(stop);
    layout->addWidget(quit);
    setLayout(layout);
    ...
}
```

Итоговое QT-приложение: camserver.cpp – слоты

Вся содержательная часть:

```
void CamServerWidget::grab() {
    grabTimer->setInterval(6000);
    QProcess v4l2grab;
    v4l2grab.start("./v4l2grab -d /dev/video0 -o image.jpg");
    v4l2grab.waitForFinished(2000);
    QProcess curl;
    curl.start("curl -F submitname=OK -F filename=@image.jpg http://server.com/upload.php");
    curl.waitForFinished(3000);
}
```

Только для мгновенного запуска таймера:

```
void CamServerWidget::start() {
    grabTimer->start(0);
}
```

Итоговое QT-приложение: сборка

- Устанавливаем зависимости

  $> *yum install make gcc-c++ qt-devel*
- Собираем прямо в MeeGo

  $> *qmake*

  $> *make*
- Запускаем

  $> *export DISPLAY=:0*

  $> *./camserver*

Итоговое QT-приложение: результат

Start grabbing at 01:38:05.654

Grabbing take 1690 msec

Uploading take 2634 msec

Start grabbing at 01:38:11.655

Grabbing take 1196 msec

Uploading take 1254 msec

Start grabbing at 01:38:17.656

Grabbing take 1685 msec

Uploading take 1299 msec

...

Домашние задания

1. Интегрировать v4l2grab в QT-приложение
2. Создать QT-обертку для v4l2grab
3. Отказаться от v4l2grab и использовать gStreamer
4. Интегрировать libcUrl в QT-приложение
5. Отказаться от cUrl/libcUrl и использовать QT Network
6. Добавить в QT-приложение окно с журналом событий и текущим кадром
7. Создать консольную версию QT-приложения, способную работать без XWindows и управляемую из командной строки

Вопросы?

БПЛА konstantinamelin@gmail.com

Приложение avkoryavko@gmail.com