Unit 5: Prototyping Connected (Emedded) Devices for the Internet of Things/ M2M

By-Er. Ishwar Rathod
SCSIT, SUAS, Indore.

Agenda

- 1) Trends of implementation of IoT applications
  - REST
  - Cloud
- 2) Connected-device Prototyping Tools
  - Arduino
  - Raspberry Pi
  - Gadgeteer
- 3) Building Web-Connected Devices With Gadgeteer
  - #1. A simple camera
  - #2. A simple Internet webcam
  - #3. A sophisticated Web-controlled camera
  - #4. Logging sensor data using Cloud-based storage
  - #5. OCR using cloud-based processing
- 4) Comparison and Recommendation

Introduction

- IoT vision
  - Internet connectivity extends to the very simplest electronic devices (things)(with a IPv6 address?)
- Advantages with IoT
  - Our productivity can be improved when the things involved in our daily
    activities become “alive” or manageable.
- Types of IoT
  - Networked versions of commonplace devices
    - Refrigerators, TV, toaster, alarm clocks, doorbells, and so on
  - Embedded devices
    - Allow applications with simple electronic devices.
    - Numerous kinds of product exist!

1) Trends of implementation of IoT applications

- Using REST as a command protocol for web-to-serial applications
  - Cloud-based service (REST-ful)

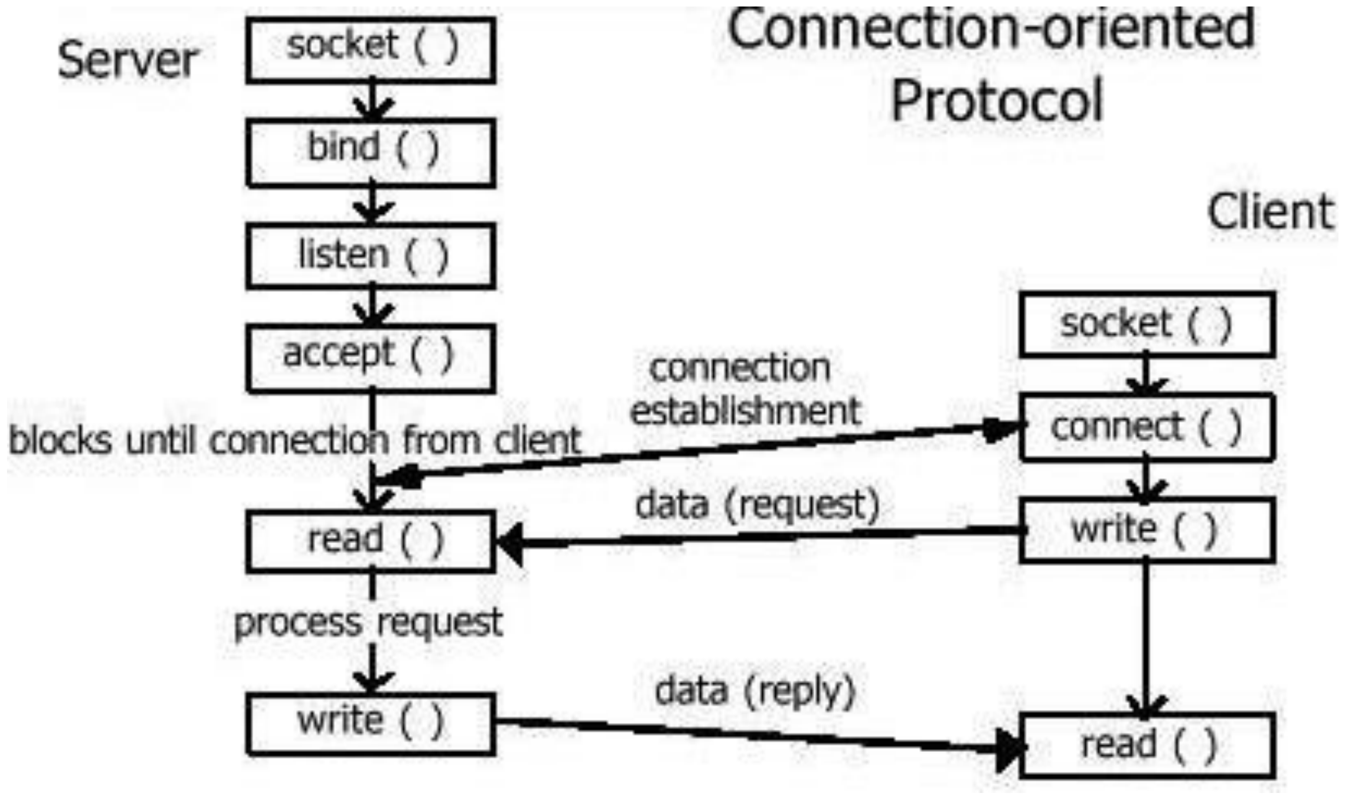
Cloud-based Web services

- The software running within the embedded device will increasingly be complemented by cloud-based Web services.
- Dramatically extend the effective processing and storage capabilities of these connected devices
- New class of applications might emerge
  - Groups of device act as the I/O elements of potentially global-scale distributed services and applications.

Cloud-based Web services

Using REST as a command protocol for web-to-serial applications

- Socket programming



Socket = IP + port

Port = 80
Write “Get HTTP..”

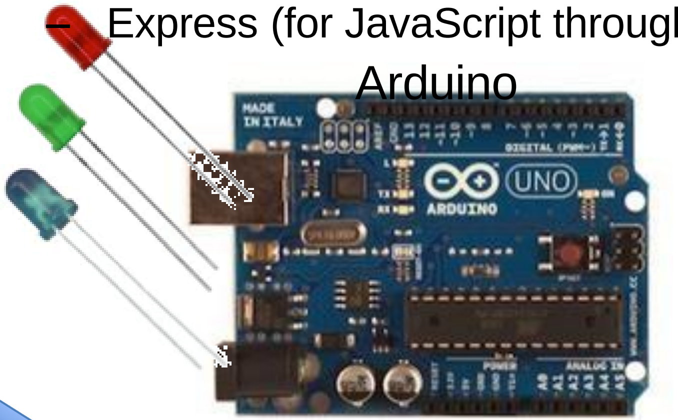
Port = 25
Write “HELO…MAIL…”

Using REST as a command protocol for web-to-serial applications

- Representational State Transfer (REST)
- Conventional use URL
  - A URL refers to a particular document on a server.
    - http://www.ouhk.edu.hk/prospectus.html
- REST (Representational State Transfer)
  - A URL is used to **set or get the state** of web-based application.
  - To get the price of item 3045:
    - http://www.mystore.com/item/3045/price
  - To set the price of item 3045:
    - http://www.mystore.com/item/3045/price/4.99

Using REST as a command protocol for web-to-serial applications

- Existing Web frameworks make it possible for you to build a web server (in your embedded devices) that uses REST as the control protocol for your application.
  - Sinatra (for Ruby),
  - Flask (for Python) and
  - Express (for JavaScript through node.js)



Firefox Start
Google
Google Search
Get the newest version! Firefox has been released. Learn more.

Web server and controller

node.js platform with express.js library

sets the level of the LEDs (range from 0 to 100)

http://200.200.200.1/LED/r/18
http://200.200.200.1/LED/g/28
http://200.200.200.1/LED/b/8

2) Introduction to Connected-device Prototyping Tools

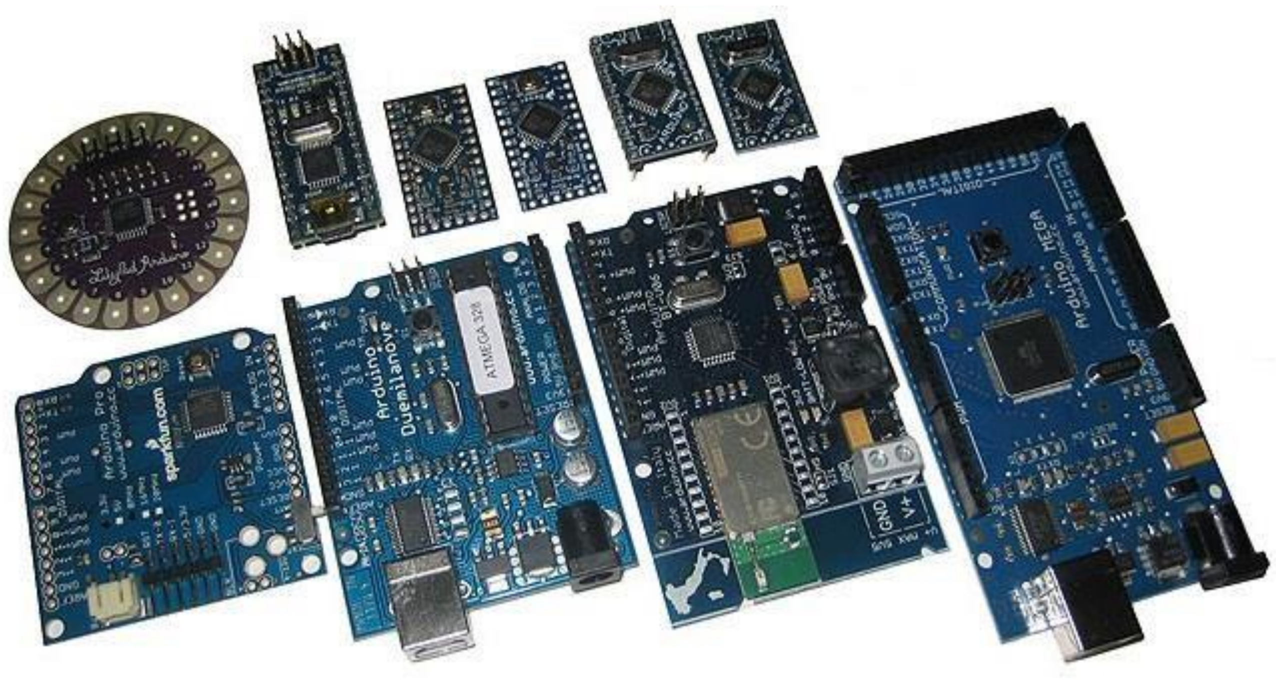
Arduino

Raspberry Pi

Gadgeteer

Arduino platform

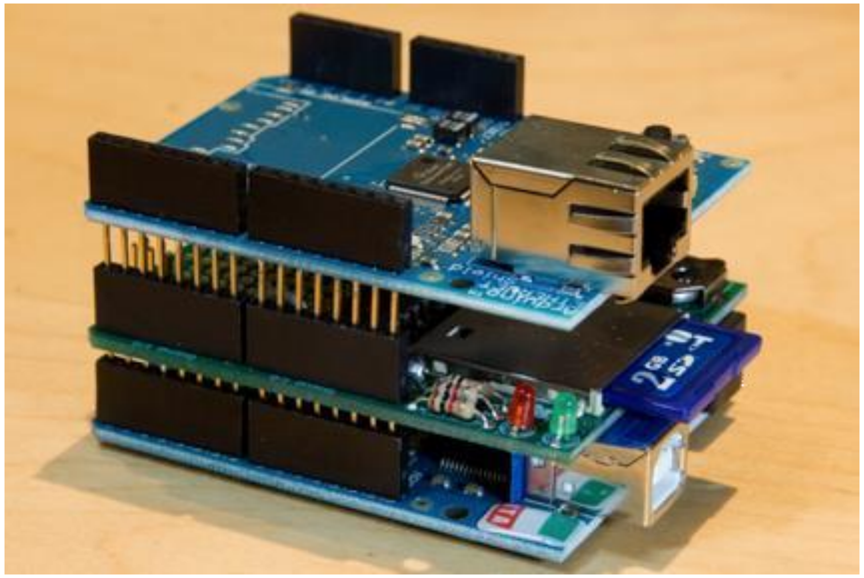
- **What**
  - Microcontroller-based platforms
  - Started in 2005 in Italy, nowadays rather presenting a family of microcontroller boards rather than a specific one.
  - An internet magazine article from May 2011, stating that there were “about 300,000+ Arduino ‘in the wild’ at that time.



Arduino platform

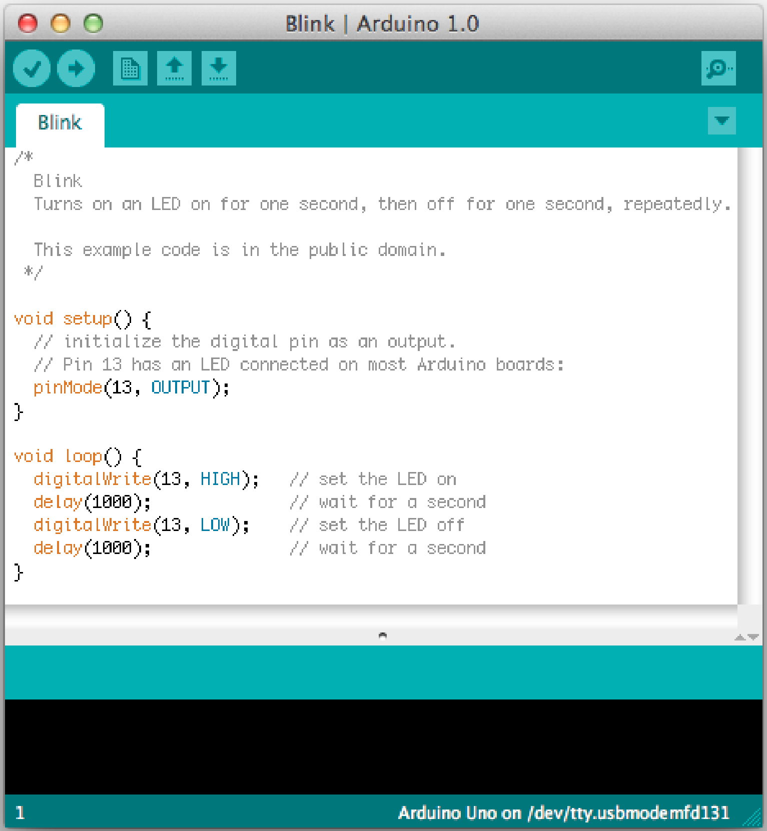
- **Hardware**
  - Shields -- add-on circuit boards that extend the platform’s basic capabilities
  - Shields that provide Ethernet, Wi-Fi, and GPRS connectivity enable Arduino’s use for connected-device development.

Arduino Uno



Arduino platform

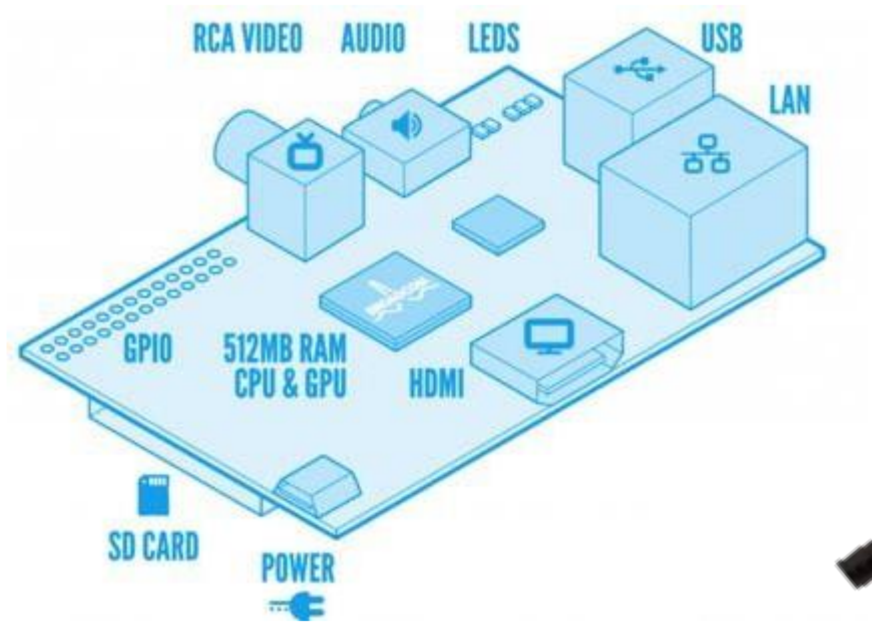
- **IDE**
  - Minimalist integrated development environment (IDE)
  - Typically programmed with C
- **Debugging**
  - Debugging is typically supported via simple communications over a serial line interface
- **Software**
  - Developers commonly use the REST technique
    - because it is a lightweight, easy-to-debug way to communicate between connected devices
  - With REST, services are exposed and accessed using HTTP, [which is readily supported by Arduino libraries]() that implement the relevant networking protocols and enable simple webserver operation.



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

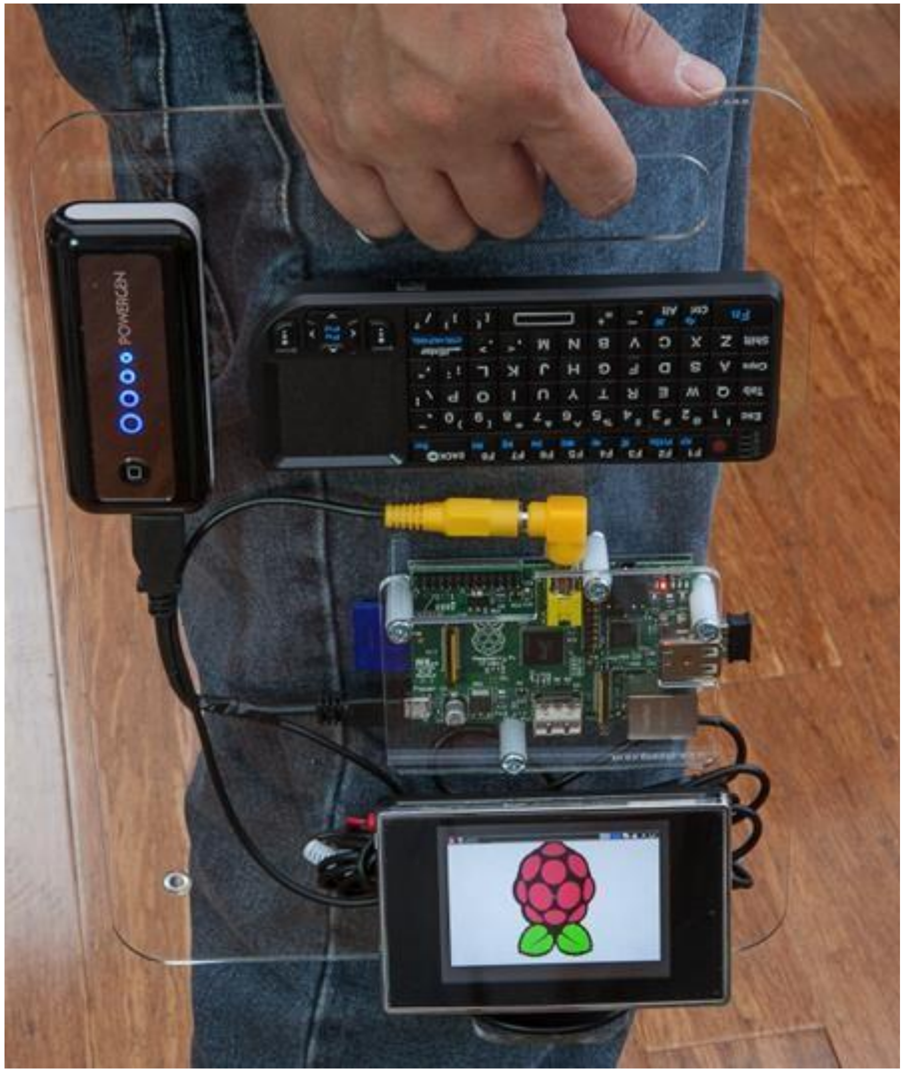
Raspberry Pi

- What
  - Small-form-factor Linux devices
  - Can be used for many of the things that your desktop PC does, like spreadsheets, word-processing, games, and playing high-definition video.



Raspberry Pi

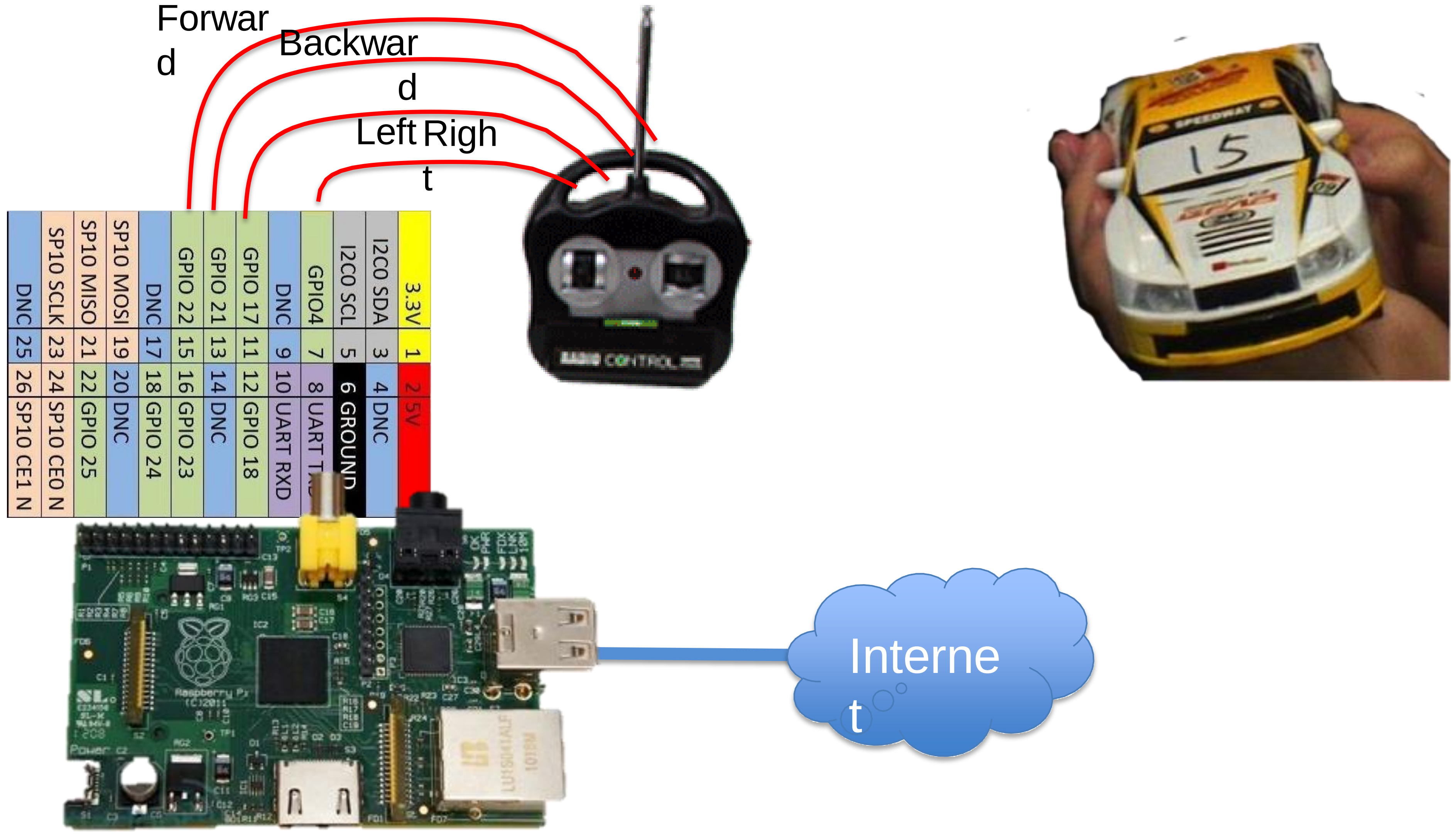
- It uses standard off the shelf hardware.
  - The LCD is a low cost TFT monitor used in car reverse camera.
  - The battery pack is a standard portable USB charger.
  - Common wireless keyboard



Raspberry Pi

Course : Integrated Project (Year 1)
Program: BEng Electronic and Computer Engineering

Raspberry Pi



3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART T
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Raspberry Pi

- Good
  - Offer the opportunity to leverage an extensive set of pre- existing tools and software components such as Node.js (http://nodejs. org),
    - which simplifies the implementation of REST-like asynchronous Web-based application programming interfaces (APIs).
  - Powerful and flexible
  - Fairly inexpensive: an ARM Linux box for US$25!
- Bad
  - Expose more complexity to the user
  - Typically less cost-effective than Arduino for lightweight device development.

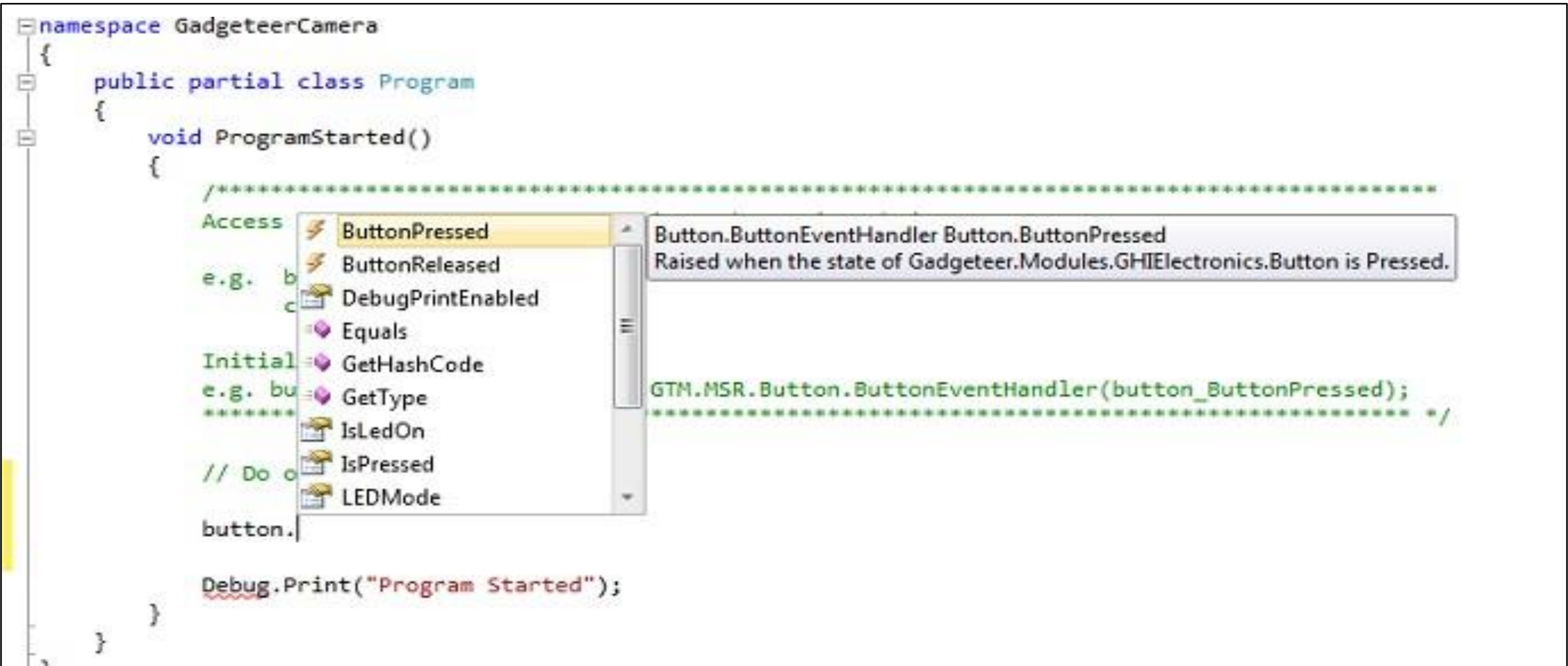
Microsoft .NET Gadgeteer

- **What**
  - fairly new, started in 2011 by Microsoft Research labs
  - An toolkit for building small electronic devices
    - using the .NET Micro Framework and Visual Studio/Visual C# Express.

- **Hardware**
  - a central “mainboard” containing a CPU and several sockets
  - a large number of different modules.

Microsoft .NET Gadgeteer.

- **IDE**
  - Tightly integrated with the Microsoft Visual Studio IDE
    - With features of dynamic syntax checking and continually provides hints and prompts to ease coding.
    - Support debugging via breakpoints, single stepping, variable watches, and execution traces.



```
namespace GadgeteerCamera
{
    public partial class Program
    {
        void ProgramStarted()
        {
            /*******************************************************************************
            Access
            e.g.  b
                  c
            Initial
            e.g. bu                               GTM.MSR.Button.ButtonEventHandler(button_ButtonPressed);
            *******                               ********************************************* */

            // Do o

            button.

            Debug.Print("Program Started");
        }
    }
```

ButtonPressed
ButtonReleased
DebugPrintEnabled
Equals
GetHashCode
GetType
IsLedOn
IsPressed
LEDMode

Button.ButtonEventHandler Button.ButtonPressed
Raised when the state of Gadgeteer.Modules.GHIElectronics.Button is Pressed.

Microsoft's Gadgeteer Design Choices

Primary design goal to <u>simplify application development as much as possible</u>

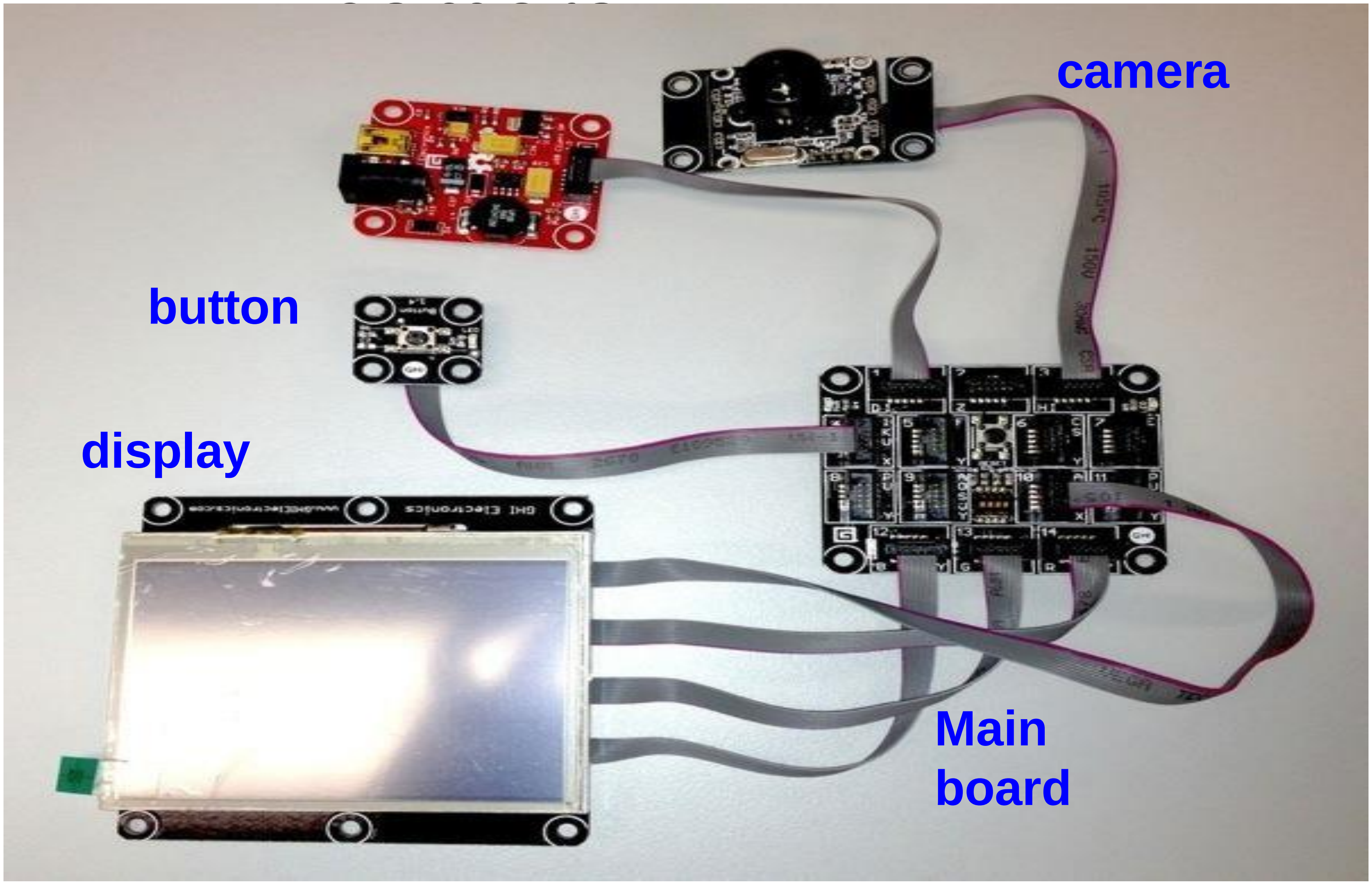
- Prioritized REST-ful support over other Web-related functionality
- Event-based model
  - Simplify the creation of many applications
  - Helps developers familiar with event-based programming on desktop and mobile platforms to transition to embedded device development.
- High-level API
  - Allow modules to be used in sophisticated ways with a few lines of code.

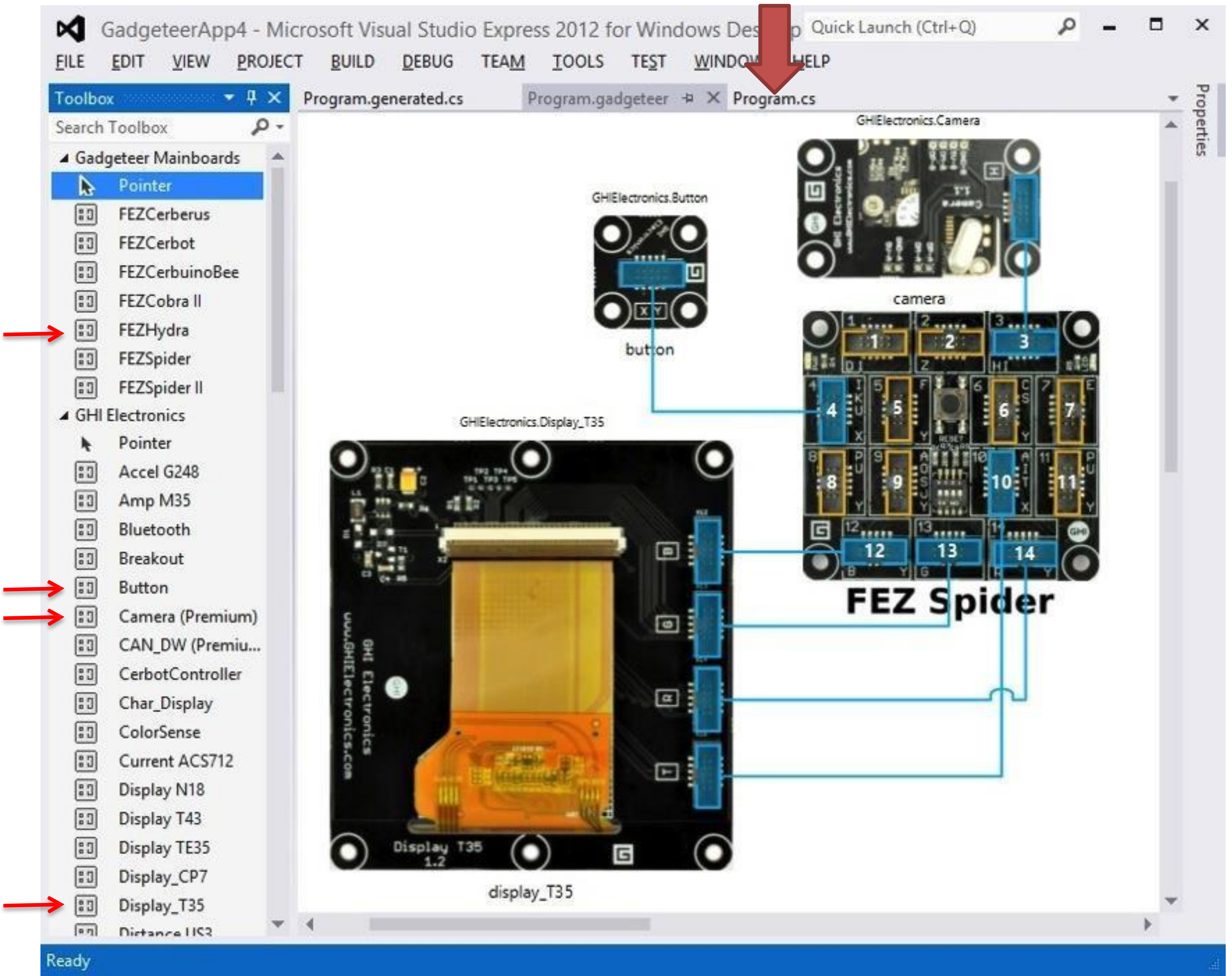
3) Building Web-Connected Devices With Gadgeteer

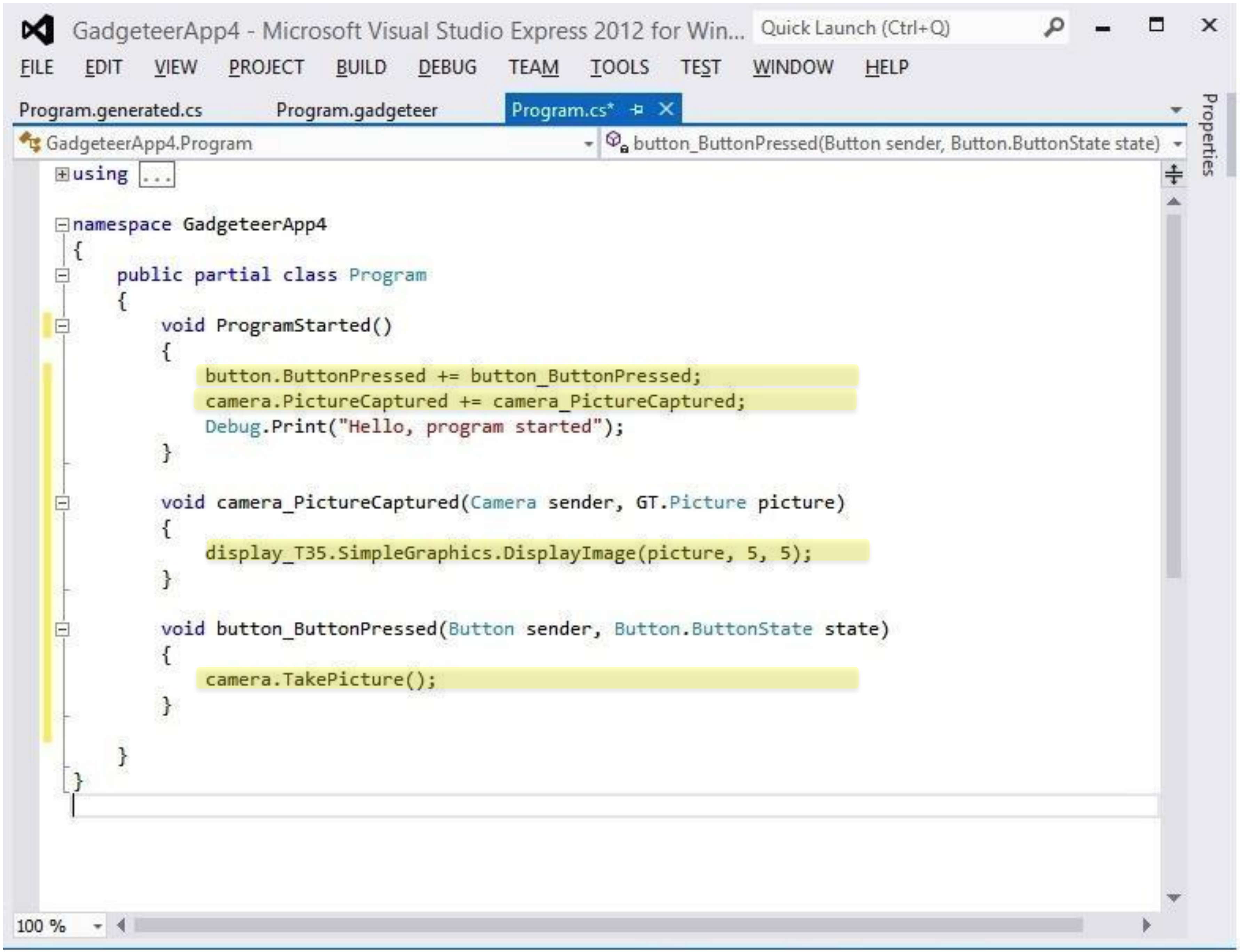
#1. A simple camera
#2. A simple Internet webcam
#3. A sophisticated Web-controlled camera
#4. Logging sensor data using Cloud-based storage
#5. OCR using cloud-based processing

Credit: #2-#5 are shown in:
Steve Hodges, et al. "Prototyping Connected Devices for the Internet of Things." *Computer*, Vol.46, Iss.2, Feb. 2013.

#1. A simple



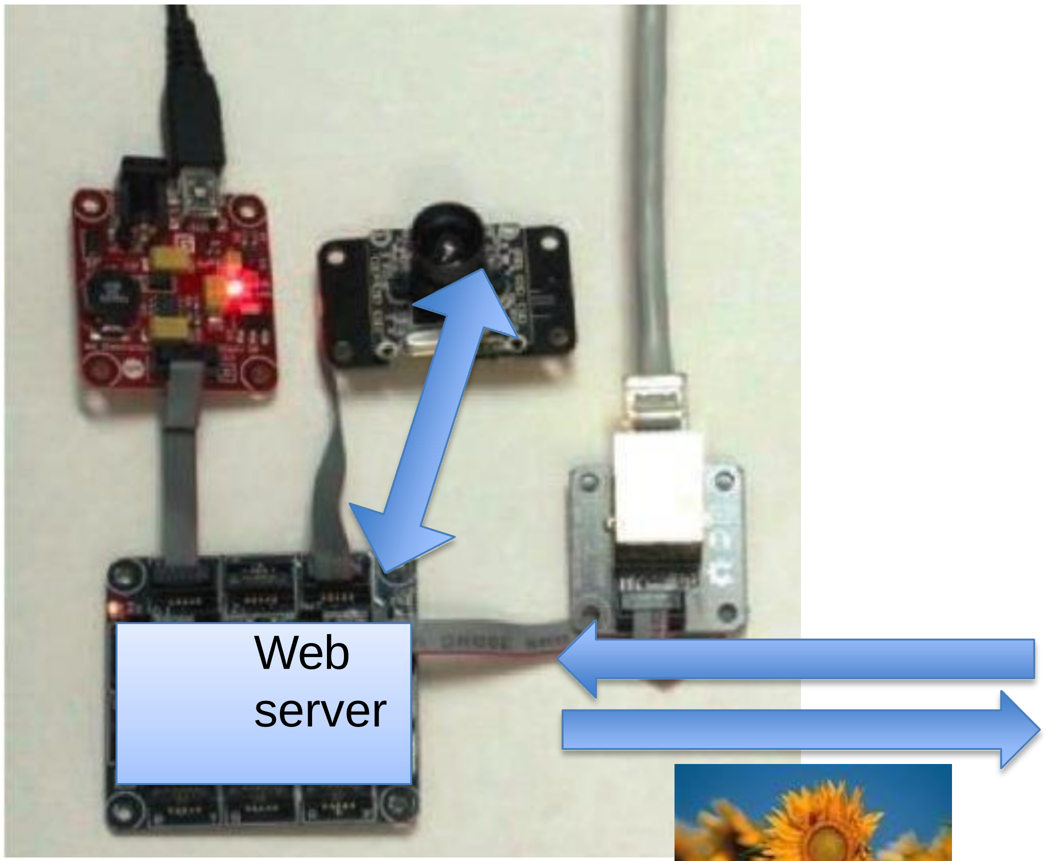

GadgeteerApp4 - Microsoft Visual Studio Express 2012 for Windows Desktop
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Toolbox
Search Toolbox
Program.generated.cs
Program.gadgeteer
Program.cs
Gadgeteer Mainboards
Pointer
FEZCerberus
FEZCerbot
FEZCerbuinoBee
FEZCobra II
FEZHydra
FEZSpider
FEZSpider II
GHI Electronics
Pointer
Accel G248
Amp M35
Bluetooth
Breakout
Button
Camera (Premium)
CAN_DW (Premiu...
CerbotController
Char_Display
ColorSense
Current ACS712
Display N18
Display T43
Display TE35
Display_CP7
Display_T35
GHIElectronics.Camera
camera
GHIElectronics.Button
button
GHIElectronics.Display_T35
display_T35
FEZ Spider
Properties
Ready


GadgeteerApp4 - Microsoft Visual Studio Express 2012 for Win...
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST WINDOW HELP
Program.generated.cs
Program.gadgeteer
Program.cs*
GadgeteerApp4.Program
button_ButtonPressed(Button sender, Button.ButtonState state)
Properties
100 %

```
using ...

namespace GadgeteerApp4
{
    public partial class Program
    {
        void ProgramStarted()
        {
            button.ButtonPressed += button_ButtonPressed;
            camera.PictureCaptured += camera_PictureCaptured;
            Debug.Print("Hello, program started");
        }

        void camera_PictureCaptured(Camera sender, GT.Picture picture)
        {
            display_T35.SimpleGraphics.DisplayImage(picture, 5, 5);
        }

        void button_ButtonPressed(Button sender, Button.ButtonState state)
        {
            camera.TakePicture();
        }

    }
}
```

#2. A simple “Internet webcam”

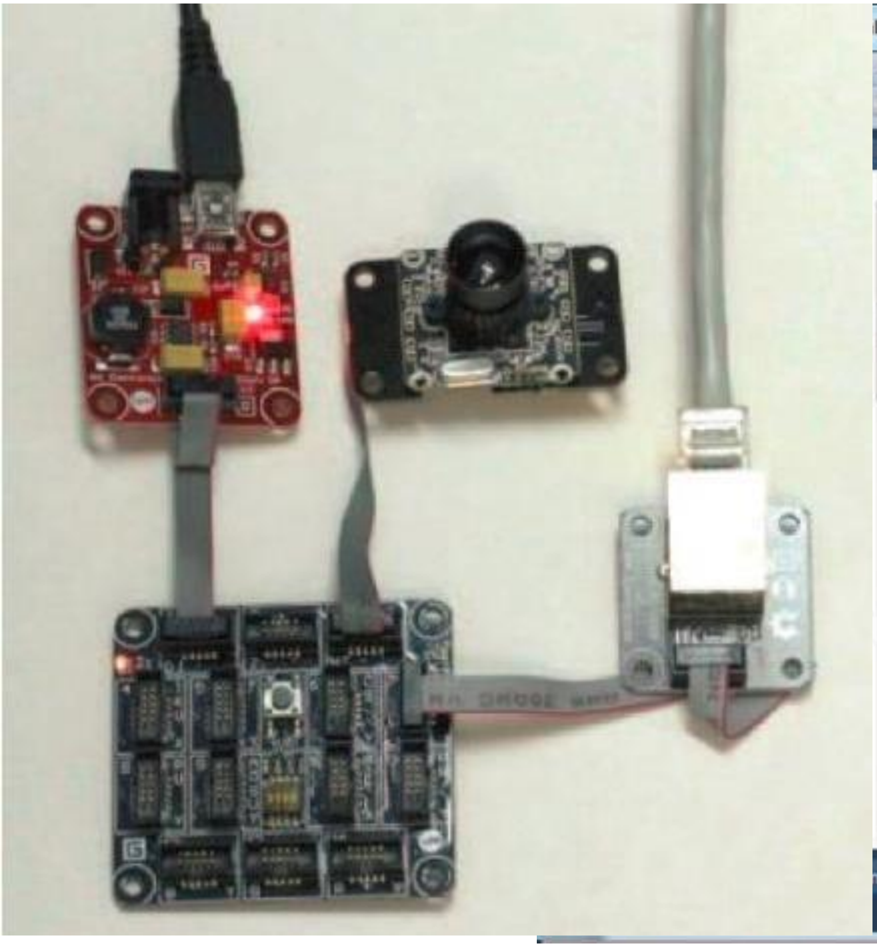
- An HTTP request from a remote client triggers the capture of a new image. The captured image is returned to the Web client.

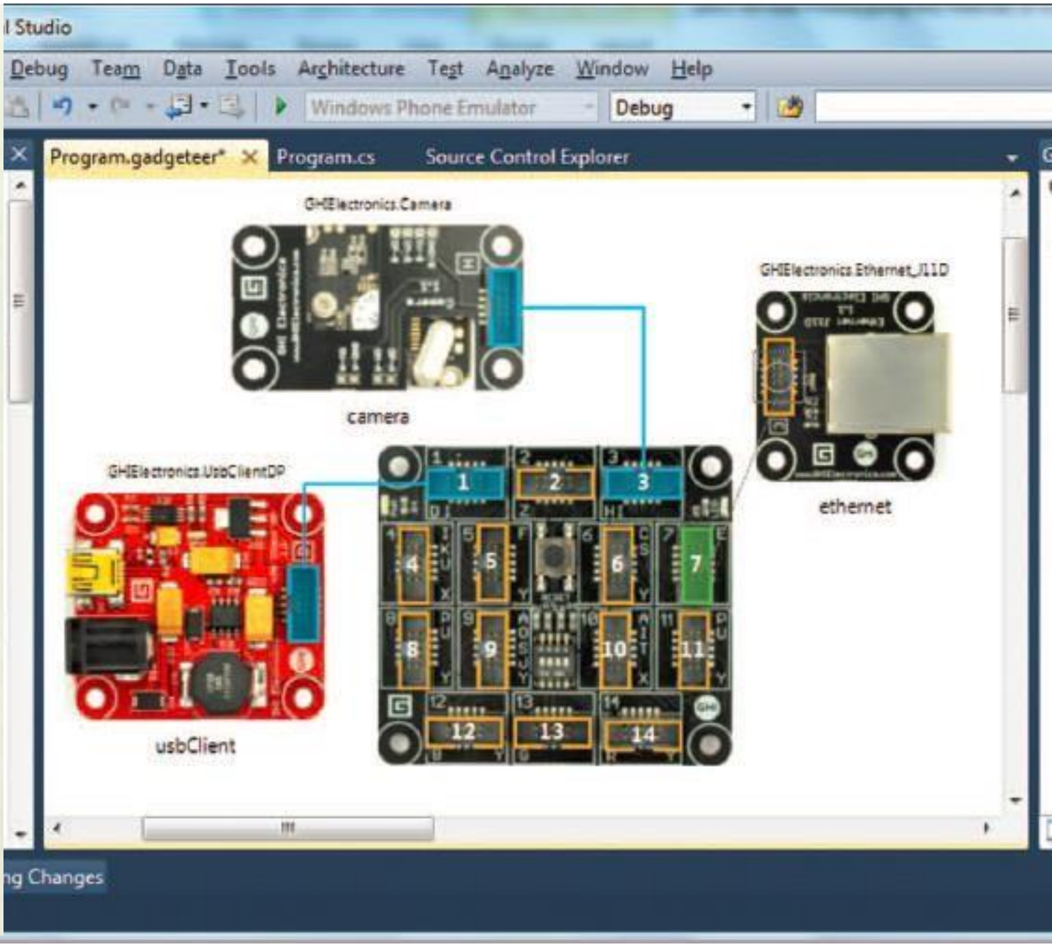


Mozilla Firefox Start Page - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Getting Started Latest Headlines
Firefox Start
Google
Web Images Groups News Local
Advanced Search
Preferences
Google Search
Firefox Help & Add-ons
About Mozilla
Mozilla Store
Get Involved
Done

Get the newest version! Firefox has been released. Learn more.

http://200.200.200.1/picture



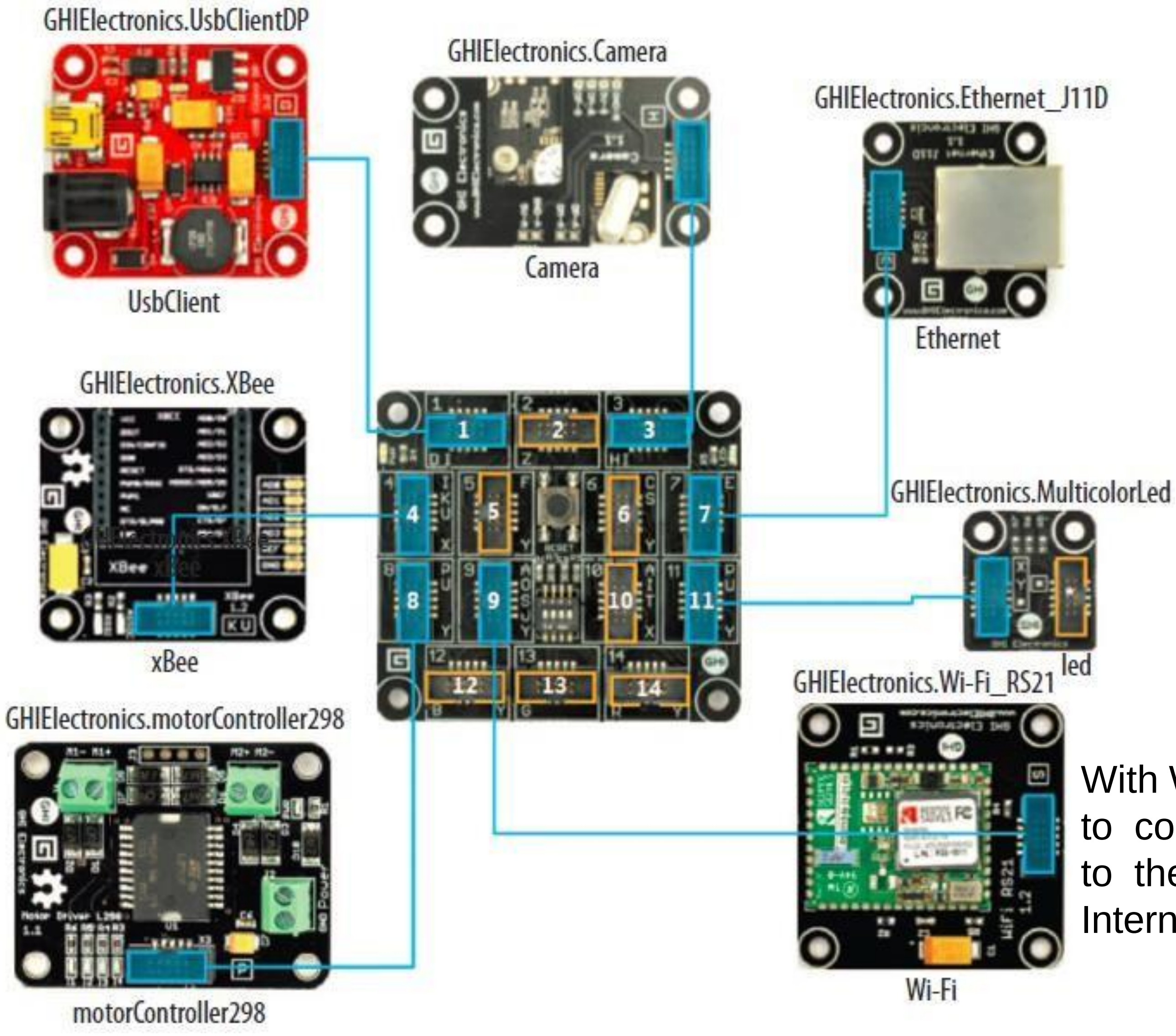

l Studio
Debug Team Data Tools Architecture Test Analyze Window Help
Windows Phone Emulator
Debug
Program.gadgeteer*
Program.cs
Source Control Explorer
GHIElectronics.Camera
camera
GHIElectronics.Ethernet_J11D
ethernet
GHIElectronics.UsbClientDP
usbClient
1
2
3
4
5
6
7
8
9
10
11
12
13
14
ng Changes

camera

ethernet

```
WebEvent cameraWebEvent;
Responder currentResponder;

void ProgramStarted()
{
      // associate PictureCaptured event with its handler
4     camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);

      // request DHCP address and associate handler for network setup
1     ethernet.UseDHCP();
      ethernet.NetworkUp += new NetworkModule.NetworkEventHandler(ethernet_NetworkUp);
}

void ethernet_NetworkUp(GTM.Module.NetworkModule sender,
        GTM.Module.NetworkModule.NetworkState state)
{
      // start a webserver on port 80
2     WebServer.StartLocalServer(ethernet.NetworkSettings.IPAddress, 80);

      // set up a handler for http '/picture' requests
3     cameraWebEvent = WebServer.SetupWebEvent("picture");
      cameraWebEvent.WebEventReceived += new
        WebEvent.ReceivedWebEventHandler(cameraWebEvent_WebEventReceived);
}

void cameraWebEvent_WebEventReceived(string path, WebServer.HttpMethod method,
        Responder responder)
{
      // initiate a picture and cache the responder to use when the picture is captured
      currentResponder = responder;
3     camera.TakePicture();
}

void camera_PictureCaptured(GTM.GHIElectronics.Camera sender, GT.Picture picture)

{
      // respond to web request with the picture
4     currentResponder.Respond(picture);
}
```

#3 A more sophisticated Web-controlled camera

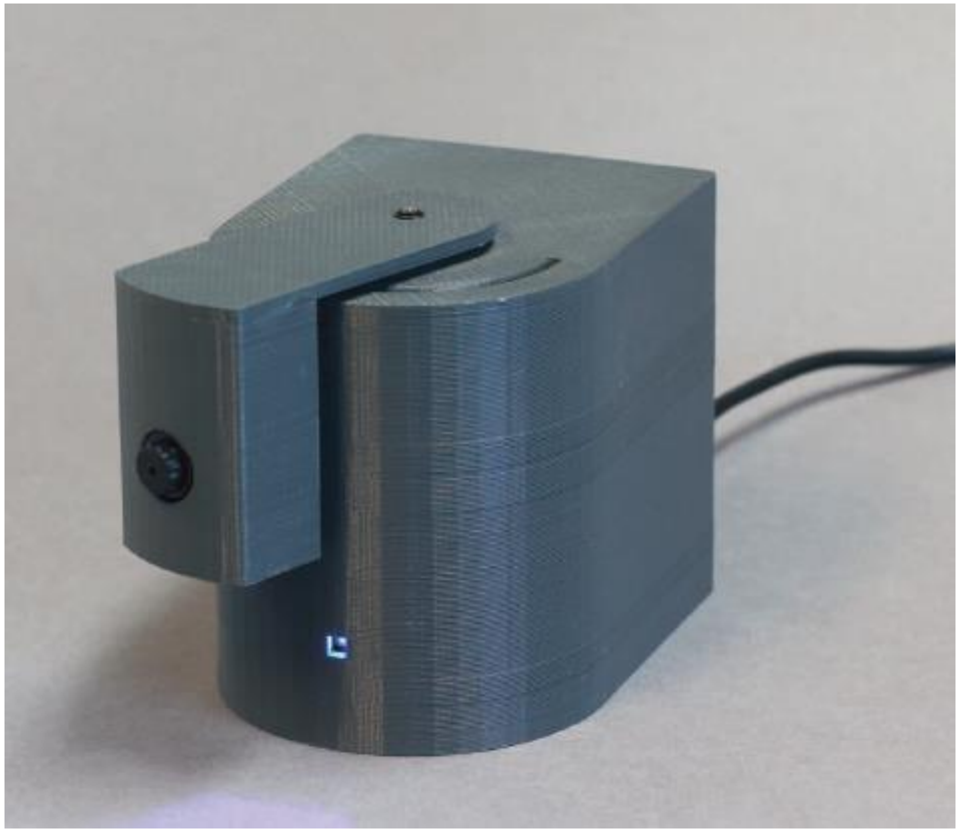

GHIElectronics.UsbClientDP
UsbClient
GHIElectronics.Camera
Camera
GHIElectronics.Ethernet_J11D
Ethernet
GHIElectronics.XBee
xBee
GHIElectronics.MulticolorLed
led
GHIElectronics.Wi-Fi_RS21
Wi-Fi
GHIElectronics.motorController298
motorController298
1
2
3
4
5
6
7
8
9
10
11
12
13
14

With Zigbee to connect to ligher-weight Gadgeteer devices e.g., temperature sensor.

With a servomotor-controlled arm, allowing remote panning as well as image capture, again over a REST-ful interface.

With Wi-Fi to connect to the Internet

#3 A more sophisticated Web-controlled camera



With a 3D-printed plastic enclosure

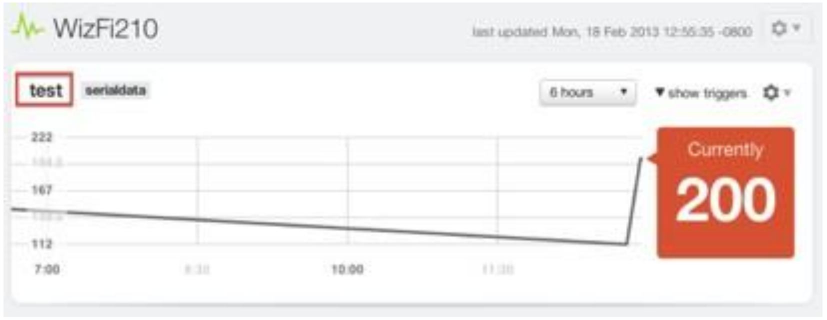
#4 Logging sensor data using Cloud-based storage

Senso
r

IoT
(Web
client)

Create a feed (POST) Updating a feed (PUT) Listing all feeds (GET) Viewing a feed (GET)

HTTP

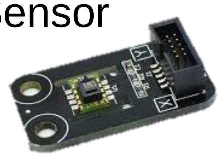


REST-ful Web
server

- A key benefit of connected operation is the potential to leverage cloud-based computation
- The Gadgeteer libraries were designed to ensure that making a Web request is as straightforward as receiving one

#4. Cloud-based processing for connected devices

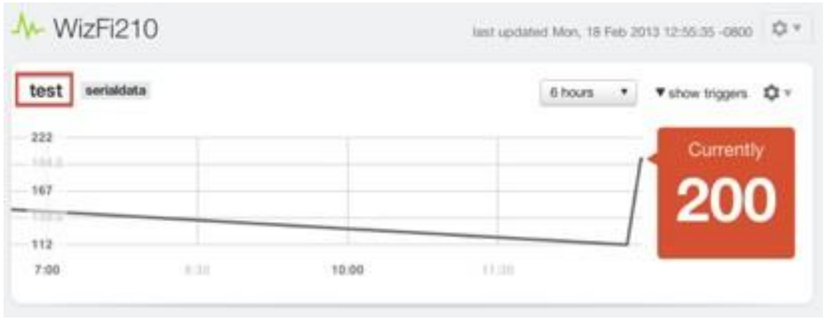
Sensor



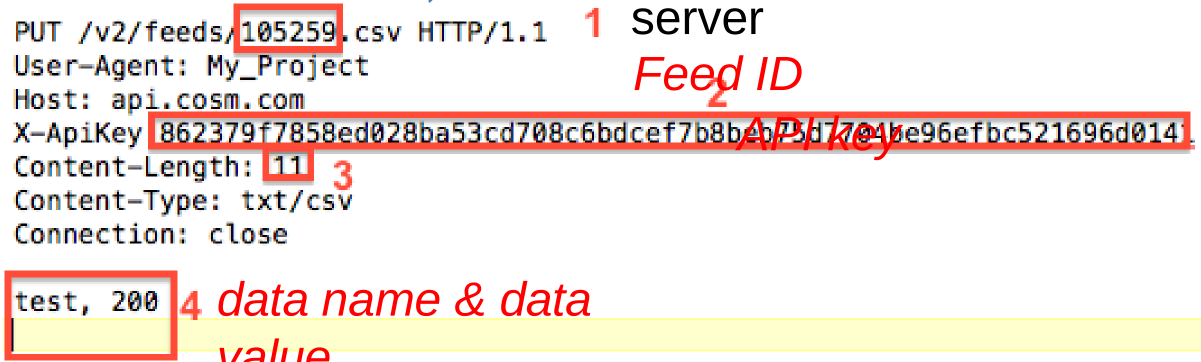
IoT
(Web client)

Create a feed (POST) Updating a feed (PUT) Listing all feeds (GET) Viewing a feed (GET)

HTTP



REST-ful Web server



```
PUT /v2/feeds/105259.csv HTTP/1.1
User-Agent: My_Project
Host: api.cosm.com
X-ApiKey 862379f7858ed028ba53cd708c6bdcef7b8beb75d7704be96efbc521696d014
Content-Length: 11
Content-Type: txt/csv
Connection: close

test, 200
```

1 *Feed ID*

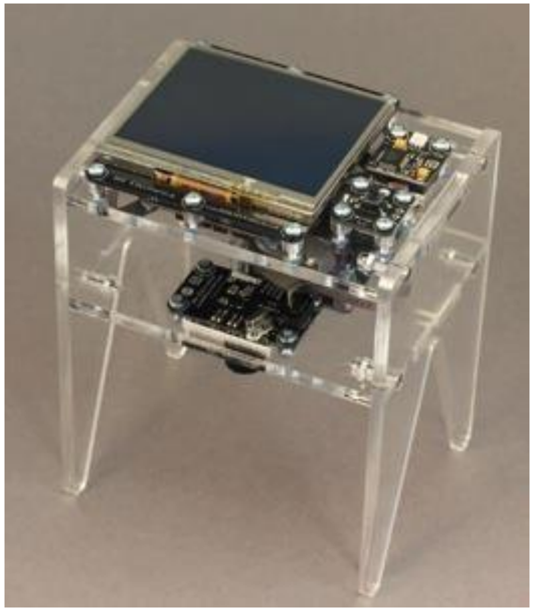
2 *API key*

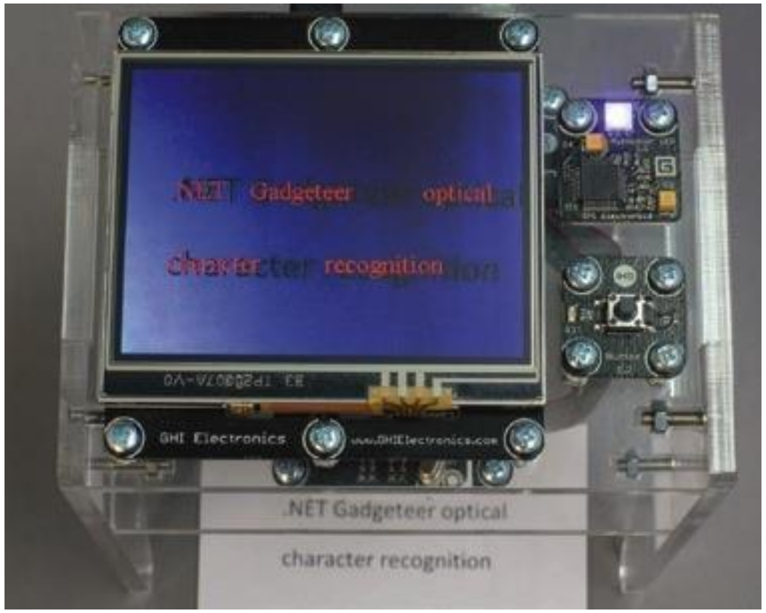
3

4 *data name & data value*

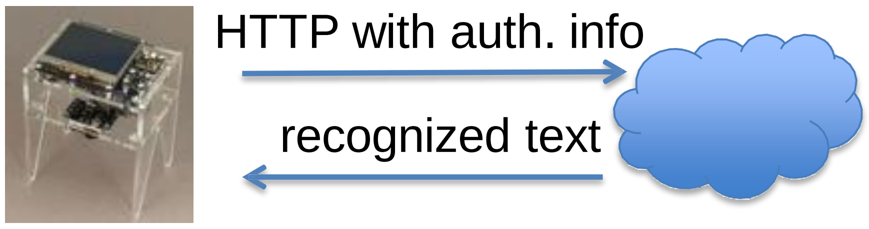
```
HttpRequest request = HttpHelper.CreateHttpPutRequest(“http://api.pachube.com/v2/fees/105259.csv”,
          PUTContent.CreateTextBasedContent(”test,”+sensorDate.Temperature.ToString()”), “text/csv”);

request.AddHeaderField(“X-ApiKey”, 862379f7858ed028ba53cd708c6bdcef7b8beb75d7704be96efbc521696d014;

request.ResponseReceoved += new Httprequest.ResponseHandler(req_ResponseReceived);

request.SendRequest();
```

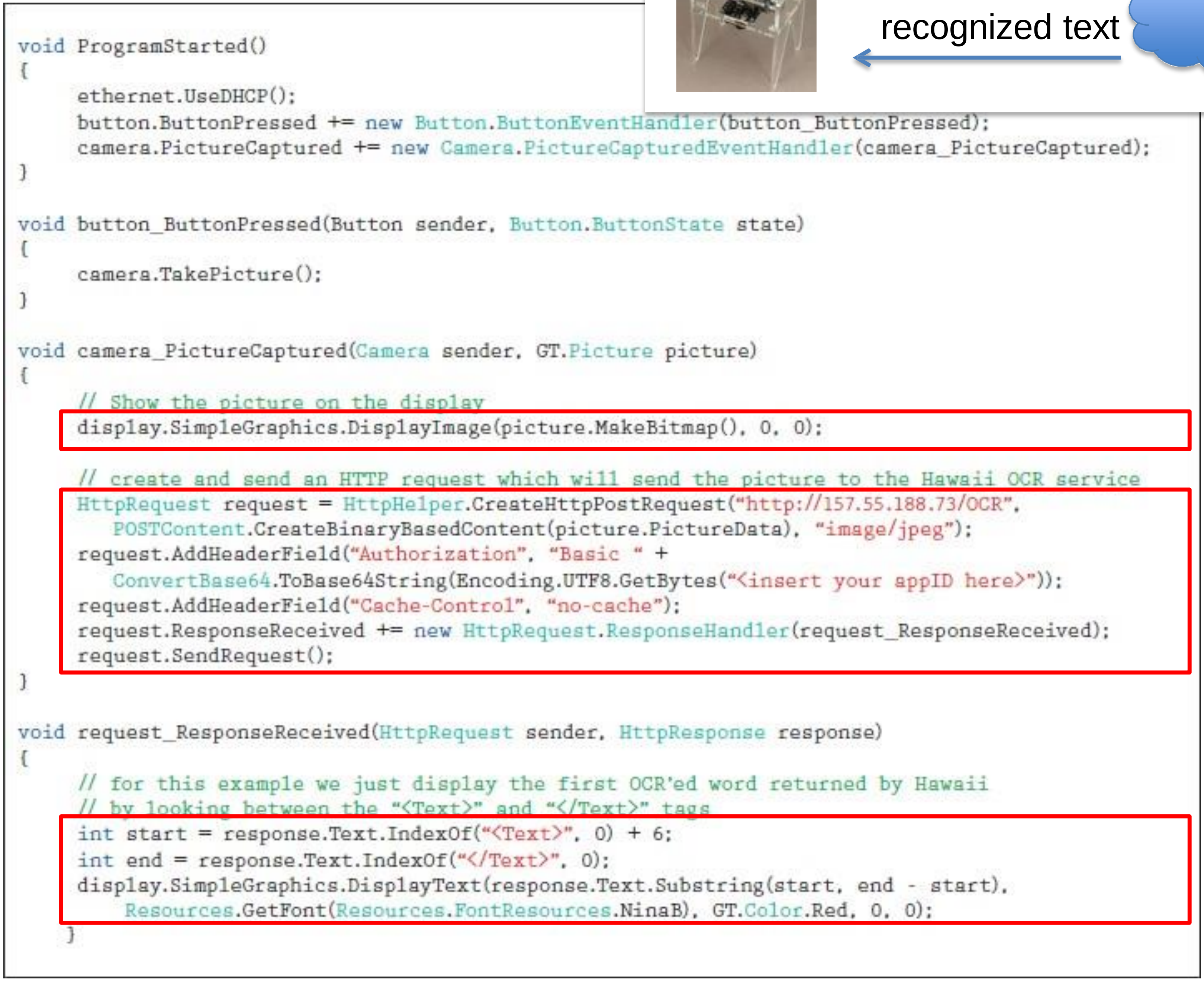
#5 OCR using cloud-based processing





- When the shutter button is pressed, it sends the image to the Project Hawaii service for OCR processing, and displays the returned text on its LCD screen





```
void ProgramStarted()
{
        ethernet.UseDHCP();
        button.ButtonPressed += new Button.ButtonEventHandler(button_ButtonPressed);
        camera.PictureCaptured += new Camera.PictureCapturedEventHandler(camera_PictureCaptured);
}

void button_ButtonPressed(Button sender, Button.ButtonState state)
{
        camera.TakePicture();
}

void camera_PictureCaptured(Camera sender, GT.Picture picture)
{
        // Show the picture on the display
        display.SimpleGraphics.DisplayImage(picture.MakeBitmap(), 0, 0);

        // create and send an HTTP request which will send the picture to the Hawaii OCR service
        HttpRequest request = HttpHelper.CreateHttpPostRequest("http://157.55.188.73/OCR",
           POSTContent.CreateBinaryBasedContent(picture.PictureData), "image/jpeg");
        request.AddHeaderField("Authorization", "Basic " +
           ConvertBase64.ToBase64String(Encoding.UTF8.GetBytes("<insert your appID here>"));
        request.AddHeaderField("Cache-Control", "no-cache");
        request.ResponseReceived += new HttpRequest.ResponseHandler(request_ResponseReceived);
        request.SendRequest();
}

void request_ResponseReceived(HttpRequest sender, HttpResponse response)
{
        // for this example we just display the first OCR'ed word returned by Hawaii
        // by looking between the "<Text>" and "</Text>" tags
        int start = response.Text.IndexOf("<Text>", 0) + 6;
        int end = response.Text.IndexOf("</Text>", 0);
        display.SimpleGraphics.DisplayText(response.Text.Substring(start, end - start),
            Resources.GetFont(Resources.FontResources.NinaB), GT.Color.Red, 0, 0);
    }
```

4) Comparison and Recommendation

	Arduino	Gadgeteer	Raspberry Pi
Model	R3	FEZ Spider	Model B
Price	$30	$120	$35
Processor	ATMega 328	ARM7	ARM11
Clock Speed	16MHz	72MHz (168MHz for FEZ Cerberus)	700MHz
RAM	2KB	16MB	256MB
Flash	32KB	4.5MB	(SD Card)
Min Power	42mA (0.3W)	160mA	700mA (3.5W)
Dev IDE	Arduino Tool	Visual Studio	IDEL, Scratch, Squeak/Linux
	Low-level C++	Managed code Concise code Real-time debugging Excellent modular concept, very good for code and hardware re-use.	being programmed in many different languages
	run one program at a time	run one program at a time	capable of running multiple programs at the same time
			Ethernet, 2USB HDMI, Composite

Recommendatns

- **For applications minimizing size -> Arduino**
  - There are very small Arduino embedded systems for making very tiny little gadget.
- **For battery powered applications -> Arduino**
  - Uses the least power of the bunch.
  - Work with a wide range of input voltages (so support a variety of different types of batteries).
- **For applications that interface to external sensors -> Arduino, Gadgeteer**
  - Arduino: lots of external (cheap) sensors
  - Gadgeteer: many slots available for connecting multiple sensors easily.
- **For applications that connect to the internet -> Gadgeteer, Raspberry Pi**
  - Gadgeteer: Full TCP/IP Stack with SSL, HTTP, TCP, UDP, DHCP
  - Raspberry Pi: the Linux OS has many components built-in that provide rather advanced networking capabilities.
- **For applications that use a graphical user interface -> Raspberry Pi**
  - It has an HDMI output.
  - A fully functional computer with graphical user interface.

Production

- No matter which tools are used for prototyping, when large-scale deployments or mass production is needed, it is more cost- effective to move to a custom PCB
  - as it can be made more cheaply and compactly through circuit integration.

Conclusions

- Different tools may be suitable for different kinds users, e.g.,
  - developers,
  - researchers,
  - designers,
  - educator, and
  - hobbyists.
- As the IoT vision gradually becomes a reality, using connected-device prototypes to explore the design space will be important.

Thanks!