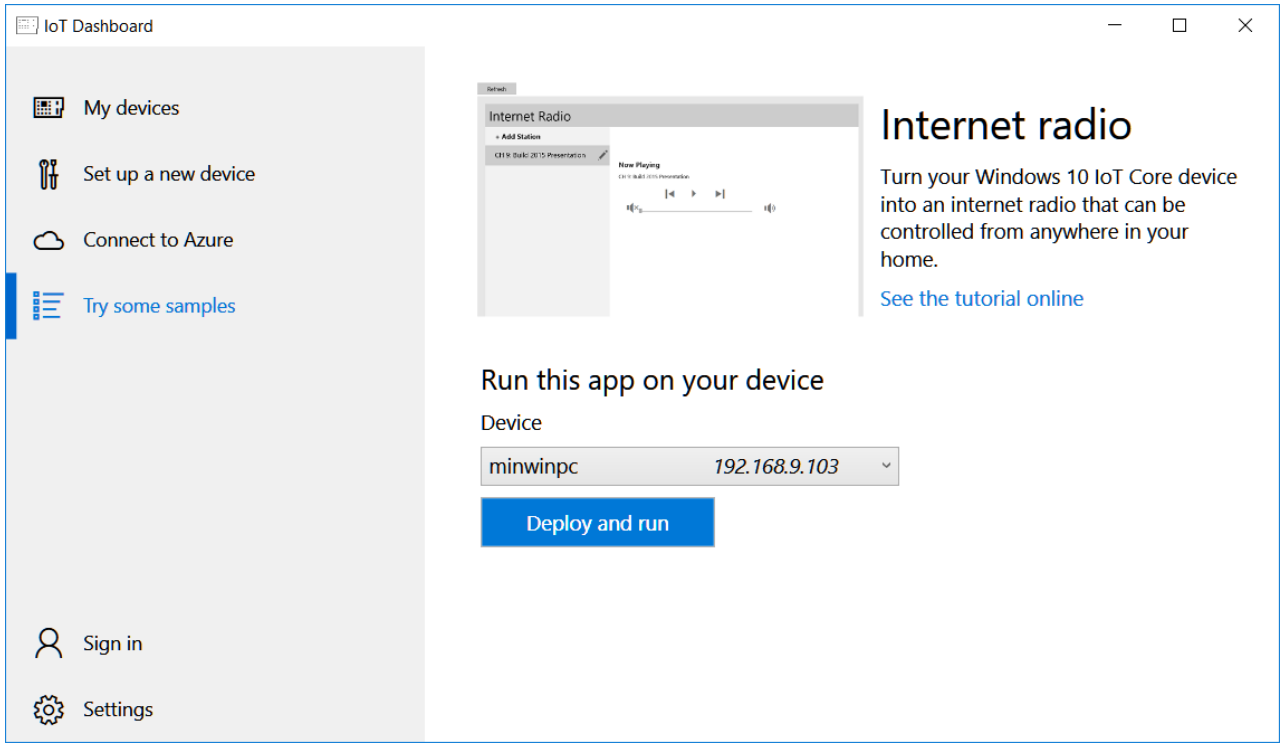
LAB 2

Windows 10 IoT Core & Visual Studio

APPLICATION DEPLOYMENT

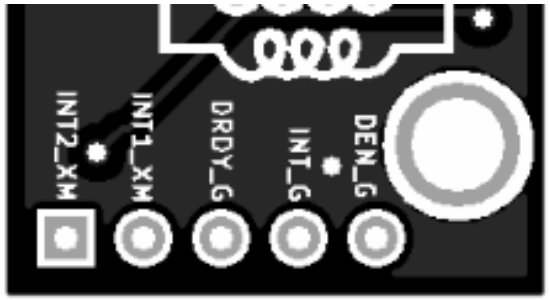
- Open IoT dashboard
- Find your device on the subnet
- Deploy and run the internet radio sample application
- Ensure you can hear channel 9



INTERRUPT CONNECTIONS

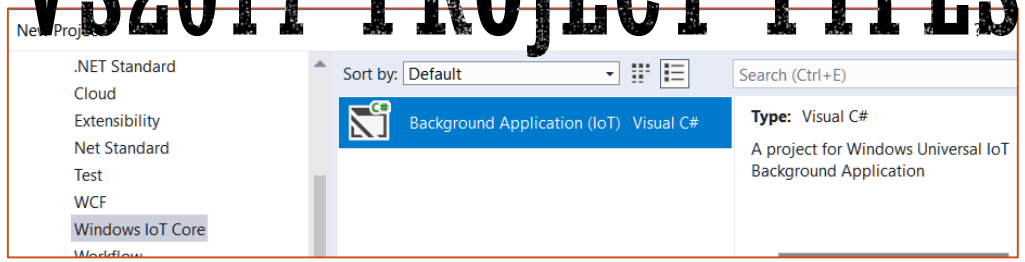
Broadcom	J8 Connector	Ozzmaker
GPIO 23	16	DRDY_M
GPIO 24	18	INT1_A_G

DRDY_M is only required for the magnetometer3 application. We will not try this application in class.



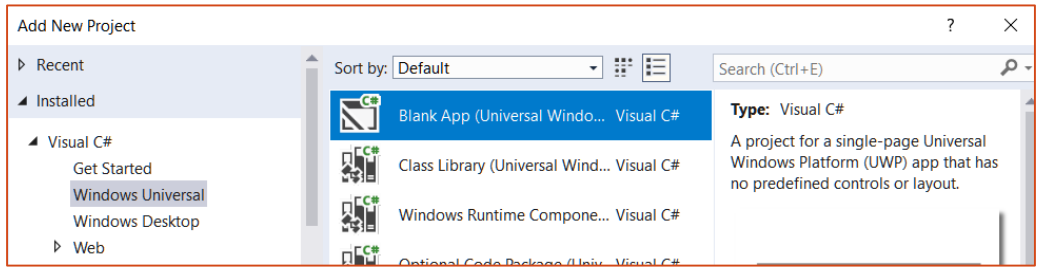
Your board is a different revision & layout

VS2017 PROJECT TYPES



Background Application

- UWP but no UI
- `class StartupTask : IBackgroundTask`
- Package Deployment
- Declaration: Background Tasks
- Assembly executed by svchost.exe



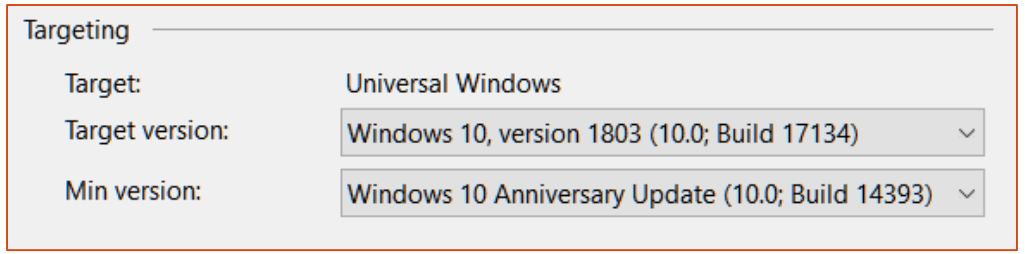
Foreground Application

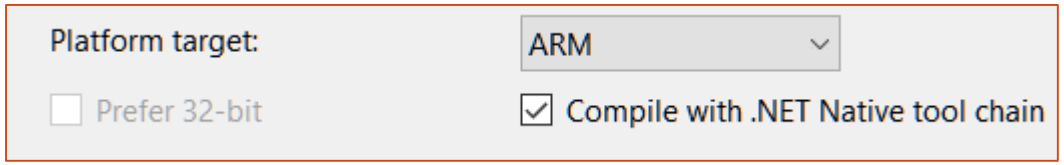
- UWP must be foreground to run
- `partial class App : Application`
- Package Deployment
- Declaration: *none*
- XAML

Choose Package->Capability->"Low Level" to use .NET GPIO libraries. Choose "Low Level Device" if you intend to write your own device driver instead. It allows IOCTLs. You may need to hand edit the declarations into the package XML.

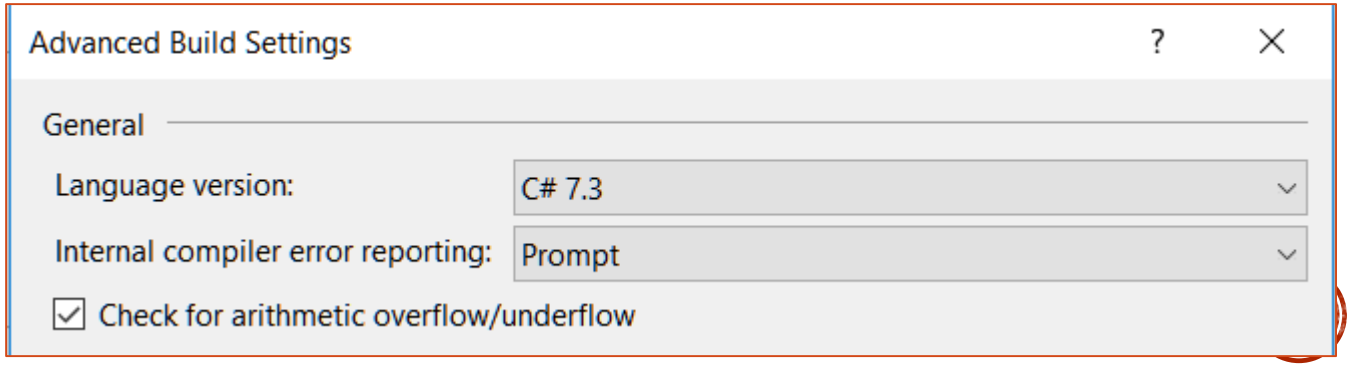
VS PROJECT SETTINGS

- Must target at least Anniversary edition to work and v1803 for full functionality
- Compile with ARM instructions
- Advanced Build Settings
  - C# 7.3 compiler has some new key words and improved code generation
  - Arithmetic checks because you will like have some math



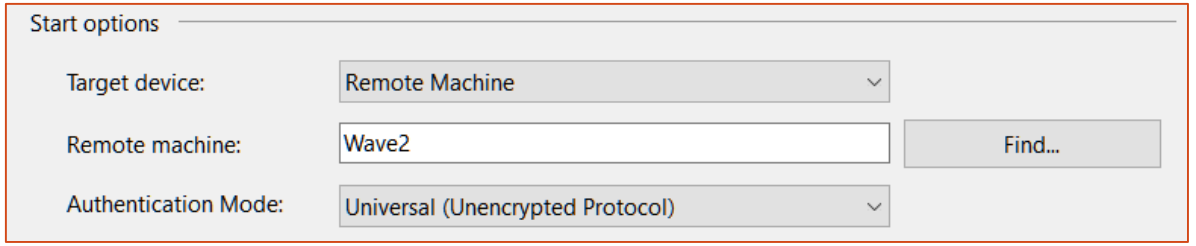


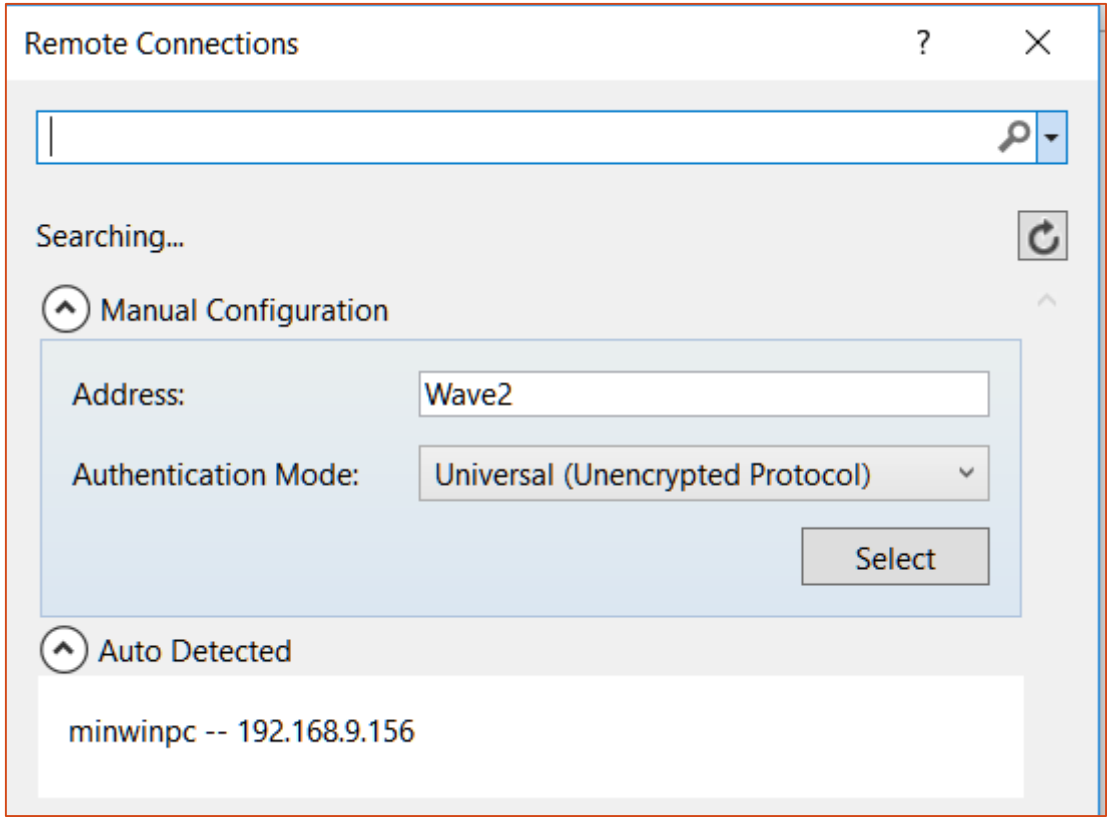
This is cross platform development



VS DEBUGGING

- Use project Deploy options to side-load code
  - Read the package file and deploys any missing sub-packages
- Use Remote Machine debugging (not Device debugging)
- Find button will search local subnet for Raspberry Pi's.
- Debugging & Deployment requires: wired VS & Pi (no Wi-Fi)





INITIALIZE I2C BUS

- Create a new Background Application IoT in VS2017 (https://docs.microsoft.com/en-us/windows/iot-core/develop-your-app/backgroundapplications)
- Add project reference to *Windows IoT Extensions for the UWP*
- Find the I2C bus
- Open the I2cDevice for the slave address
  - 400 kHz bus speed

```
// Get a selector string that will return all I2C controllers on the system
string allIc2Controllers = I2cDevice.GetDeviceSelector();

// Find the I2C bus controller device with our selector string
var discoveredI2cDevices =
     await DeviceInformation.FindAllAsync(allIc2Controllers);
var discoveredI2cDevice = discoveredI2cDevices[0].Id;

// Create I2cDevices with our selected bus controller and I2C settings
var i2cConnectionSettings = new I2cConnectionSettings(slaveAddress)
{
    BusSpeed = I2cBusSpeed.FastMode, // Enable 400kHz I2C bus speed
};

// Create I2cDevices with our selected bus controller and I2C settings
var i2cDevice = await I2cDevice.FromIdAsync(
    discoveredI2cDevice, i2cConnectionSettings);
```

UWP & EVENTS

- Create Blank UWP Application
- Add reference to *Windows IoT Extensions for the UWP*
- Select package capability *Low Level*
- Program a GPIO pin for input
  - Fire an event when signal goes high
  - Event thread taken from thread pool
  - Better performance than polling
  - May need to write device driver if time critical

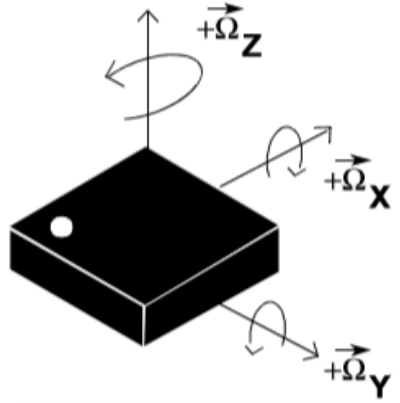
```
//fire an event when the data ready pin goes high
drdyMPin = EnableInterruptMonitor(DRDY_M,
    InterruptDrdyM,
    GpioPinDriveMode.InputPullDown);
while (ReadData().HasValue) ;


/// <summary>
/// Receive the interrupt event that
/// data is ready
/// </summary>
/// <param name="sender">the pin</param>
/// <param name="args">the event</param>
protected void InterruptDrdyM(GpioPin sender,
GpioPinValueChangedEventArgs args) {
    if (drdyMPin.Read() == GpioPinValue.High)
    {
        var newReading = ReadData();
        NewReading?.Invoke(this, newReading);
    } }
```

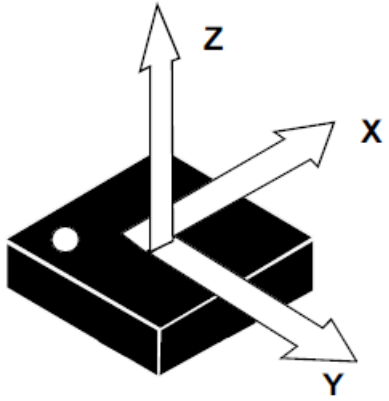
IOT APPLICATION #2: BOBBLE HEAD SANTA

- Mount Raspberry Pi on a bobble head Santa.
  - Place an inertial measurement unit in Santa's head.
  - Connect a speaker to the Pi
- Use gyroscope & accelerometer to determine when the head is bobbing.
- Play a Jingle Bells with the tempo based on the angular velocity and linear acceleration.
  - Replace with MP3 of your choice

GYROSCOPE & ACCELEROMETER



- STμ LSM9DS1, 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor
- Same BerryIMU breakout board, http://ozzmaker.com/berryimu/
- I²C slave device on address 0x6A
- WHO_AM_I register 0x0F contains 0x68
- I2C bus
  - 400 KHz, fast mode operation
- Linear acceleration full scale of ±2g/±4g/±8/±16 g
- Angular rate of ±245/±500/±2000 degrees per second.



STATUS REGISTER

- Data ready pin transitioning to high causes event to fire
- Read status register
- Determine what data is read
- Read it
- Fire event for gyroscope data
- Fire different event for accelerometer data

```
var status = (StatusRegister)ReadByteFrom(STATUS_REG);

if (status.HasFlag(StatusRegister.GDA))
{
     var gyroReading = GetGyroscopeReadings();
     NewGyroscopeReading?.Invoke(this, gyroReading);
}
if (status.HasFlag(StatusRegister.XDA))
 {
     var accelmeterReading =
        GetAccelerometerReadings();
     NewAccelerometerReading?.Invoke(this,
        accelmeterReading);
 }
if (status.HasFlag(StatusRegister.TDA))
    DiscardTemperatureData();
```

MEDIA PLAYER

- Play an mp3 file distributed with the package
- Set as the source to the player
- Alter the playback session's rate based on head motion

```
var mediaPlayer =
  new MediaPlayer();
var audioPath = new Uri(
  @"ms-appx:///JingleBells.mp3");
var source = MediaSource.
  CreateFromUri(audioPath);
mediaPlayer.Source = source;
mediaPlayer.Play();
mediaPlayer.PlaybackSession.
  PlaybackRate = value;
```

BONUS SLIDES

If time allows

GCC LINUX PI

- Compiler the settings required for most GNU tools are as follows:
  - march=armv6
  - mfpu=vfp
  - mfloat-abi=hard
- Produce code for armv6 specific instructions
  - vector floating point instructions

RASPBIAN GPIO & EVENTS

- Linux GPIO statement
  - GPIO interfaces in the kernel
- Universal File System
  - /sys/class/gpio
  - echo 23 > /sys/class/gpio/export
  - ls –lh /sys/class/gpio/gpio23
  - hardware pin can turn on interrupts by writing your desired setting into the edge file
  - Another example
- wiringPi
  - #include <wiringPi.h>
  - Arduino clone
  - Functions
  - Requires Superuser
  - Assumes ownership of all Pi hardware: SPI, IC2, GPIO, UART, etc…
- Pigpio Library
  - Requires Superuser
  - Polling, not interrupt based, uses timers
  - Several versions including a daemon

RASPBIAN REMOTE ACCESS

- SSL is turned off as installed
  - sudo service ssh start
  - sudo systemctl enable ssh
  - Then SSH will work (no GUI, of course)
- A pre-installed vnc-server
  - Configuration utility enables
  - Requires a purchased commercial license to use