

Data Collection: Sensing II

EE382V Activity Sensing and Recognition

Announcements

Project Proposal due on **Thursday Sept 29th**

Turn it in on Canvas

Text entry form or PDF



Mobile Phone

Collecting Sensor Data

CMMotionManager

Gateway to motion services

Accelerometer

Gyroscope

Magnetometer

Altitude

AVAudioSession

Record audio

AVCamCaptureManager

Take photos

Collecting Sensor Data

Handling Motion Updates at Specified Intervals

- **Accelerometer.** Set the `accelerometerUpdateInterval` property to specify an update interval. Call the `startAccelerometerUpdates(to:withHandler:)` method, passing in a block of type `CMAccelerometerHandler`. Accelerometer data is passed into the block as `CMAccelerometerData` objects.
- **Gyroscope.** Set the `gyroUpdateInterval` property to specify an update interval. Call the `startGyroUpdates(to:withHandler:)` method, passing in a block of type `CMGyroHandler`. Rotation-rate data is passed into the block as `CMGyroData` objects.
- **Magnetometer.** Set the `magnetometerUpdateInterval` property to specify an update interval. Call the `startMagnetometerUpdates(to:withHandler:)` method, passing a block of type `CMMagnetometerHandler`. Magnetic-field data is passed into the block as `CMMagnetometerData` objects.

Collecting Sensor Data

Accelerometer

```
// Track motion
self.motionManager = [[CMMotionManager alloc] init];
self.motionManager.accelerometerUpdateInterval = .2;

[self.motionManager startAccelerometerUpdatesToQueue:[NSOperationQueue currentQueue]
                                withHandler:^(CMAccelerometerData
*accelerometerData, NSError *error)
{
    dispatch_async(dispatch_get_main_queue(),
        ^{
            [self processAccelerometerData: accelerometerData.acceleration];

            if(error)
            {
                NSLog(@"%@", error);
            }
        });
}];
```

Collecting Sensor Data

Audio Recording

```
// Setup audio session
AVAudioSession* session = [AVAudioSession sharedInstance];
[session setCategory:AVAudioSessionCategoryPlayAndRecord error:nil];

// Define the recorder setting
NSMutableDictionary *recordSetting = [[NSMutableDictionary alloc] init];

[recordSetting setValue:[NSNumber numberWithInt:kAudioFormatMPEG4AAC] forKey:AVFormatIDKey];
[recordSetting setValue:[NSNumber numberWithFloat:44100.0] forKey:AVSampleRateKey];
[recordSetting setValue:[NSNumber numberWithInt: 2] forKey:AVNumberOfChannelsKey];

// Initiate and prepare the recorder
self.audioRecorder = [[AVAudioRecorder alloc] initWithURL:outputFileURL
settings:recordSetting error:nil];
[self.audioRecorder record];
[session setActive:YES error:nil];

[self.audioRecorder stop];
AVAudioSession* audioSession = [AVAudioSession sharedInstance];
[audioSession setActive:NO error:nil];
```

Collecting Sensor Data

Capture Photos

```
// Init the device inputs
AVCaptureDeviceInput *newVideoInput = [[AVCaptureDeviceInput alloc] initWithDevice:[self backFacingCamera]
error:nil];

// Setup the still image file output
AVCaptureStillImageOutput *newStillImageOutput = [[AVCaptureStillImageOutput alloc] init];
NSDictionary *outputSettings = [[NSDictionary alloc] initWithObjectsAndKeys:
    AVVideoCodecJPEG, AVVideoCodecKey,
    nil];
[newStillImageOutput setOutputSettings:outputSettings];

// Create session (use default AVCaptureSessionPresetHigh)
AVCaptureSession *newCaptureSession = [[AVCaptureSession alloc] init];

AVCaptureConnection* stillImageConnection = [AVCamUtilities connectionWithMediaType:AVMediaTypeVideo
                                                                                      fromConnections:[self
stillImageOutput] connections]];

if ([stillImageConnection isVideoOrientationSupported])
    [stillImageConnection setVideoOrientation:orientation];

[[self stillImageOutput] captureStillImageAsynchronouslyFromConnection:stillImageConnection
                                                                    completionHandler:
    ^(CMSampleBufferRef imageDataSampleBuffer, NSError *error)
    {...}
```




Microsoft Band

Collecting Sensor Data

Sensor Stream	Details	Frequency
Accelerometer	Provides X, Y, and Z acceleration in g units. 1 g = 9.81 meters per second squared (m/s ²).	62/31/8 Hz
Gyroscope	Provides X, Y, and Z angular velocity in degrees per second (°/sec) units.	62/31/8 Hz
Distance	Provides the total distance in centimeters, current speed in centimeters per second (cm/s), current pace in milliseconds per meter (ms/m), and the current pedometer mode (such as walking or running).	1 Hz
Heart Rate	<p>Provides the number of beats per minute; also indicates if the heart rate sensor is fully locked on to the wearer's heart rate.</p> <p>The data returned should be used only in resting mode. The SDK doesn't provide access to the heart rate values optimized for any other activity.</p>	1 Hz
Pedometer	Provides the total number of steps the wearer has taken since the Band was last factory-reset. This is a lifetime counter and not a daily or a 0-based counter. To determine the absolute number of steps between two readings, you must take the difference between the returned values.	Value change
Skin Temperature	Provides the current skin temperature of the wearer in degrees Celsius.	1 Hz

Collecting Sensor Data

UV	Provides the current ultraviolet radiation exposure intensity.	1 Hz
Band Contact	Provides the current state of the Band as being worn/not worn.	Value change
Calories	Provides the total number of calories the wearer has burned since the Band was last factory-reset. This is a lifetime counter and not a daily or a 0-based counter. To determine the absolute number of calories burned between two readings, you must take the difference between the returned values.	Value change
Galvanic Skin Response	(Microsoft Band 2 only) Provides the current skin resistance of the wearer in kohms.	0.2/5 Hz
RR Interval	(Microsoft Band 2 only) Provides the interval in seconds between the last two continuous heart beats. The data returned should be used only in resting mode. The SDK doesn't provide access to the RR interval values optimized for any other activity.	Value change
Ambient Light	(Microsoft Band 2 only) Provides the current light intensity (illuminance) in lux (Lumes per sq. meter).	2 Hz
Barometer	(Microsoft Band 2 only) Provides the current raw air pressure in hPa (hectopascals) and raw temperature in degrees Celsius.	1 Hz
Altimeter	(Microsoft Band 2 only) Provides current elevation data like total gain/loss, steps ascended/descended, flights ascended/descended, and elevation rate.	1 Hz

Collecting Sensor Data

```
import com.microsoft.band.sensors.BandHeartRateEvent;  
import com.microsoft.band.sensors.BandHeartRateEventListener;  
import com.microsoft.band.sensors.HeartRateConsentListener;
```

```
@Override  
public void userAccepted(boolean consentGiven) {  
    // handle user's heart rate consent decision  
};
```

```
// check current user heart rate consent  
if(client.getSensorManager().getCurrentHeartRateConsent() !=  
UserConsent.GRANTED) {  
    // user hasn't consented, request consent  
    // the calling class is an Activity and implements  
    // HeartRateConsentListener  
    bandClient.getSensorManager().requestHeartRateConsent(this,  
this);  
}
```


Collecting Sensor Data

```
// create a heart rate event listener
BandHeartRateEventListener heartRateListener = new
BandHeartRateEventListener() {
    @Override
    public void onBandHeartRateChanged(BandHeartRateEvent event) {
        // do work on heart rate changed (i.e., update UI)
    }
};
```

```
try {
    // register the listener
    bandClient.getBandSensorManager().registerHeartRateEventListener(
heartRateListener);
} catch (BandIOException ex) {
    // handle BandException
}
```

```
try {
    // unregister the listener
    bandClient.getBandSensorManager().unregisterHeartRateEventListene
r(heartRateListener);
} catch (BandIOException ex) {
    // handle BandException
}
```

Demo

Papers

Form groups of 3 or 4

Discuss what you liked/disliked about the paper

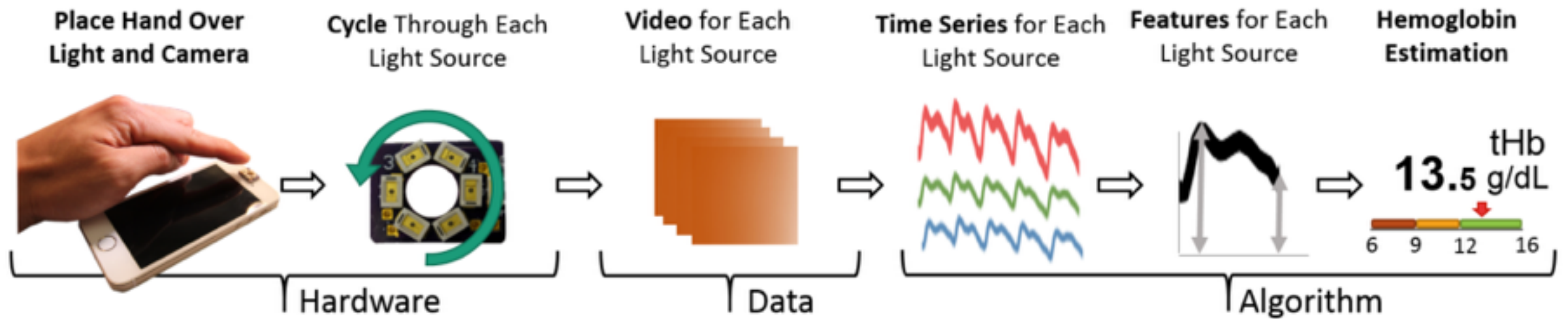
Come up with 1 question or point about each paper

10 minutes

HemaApp

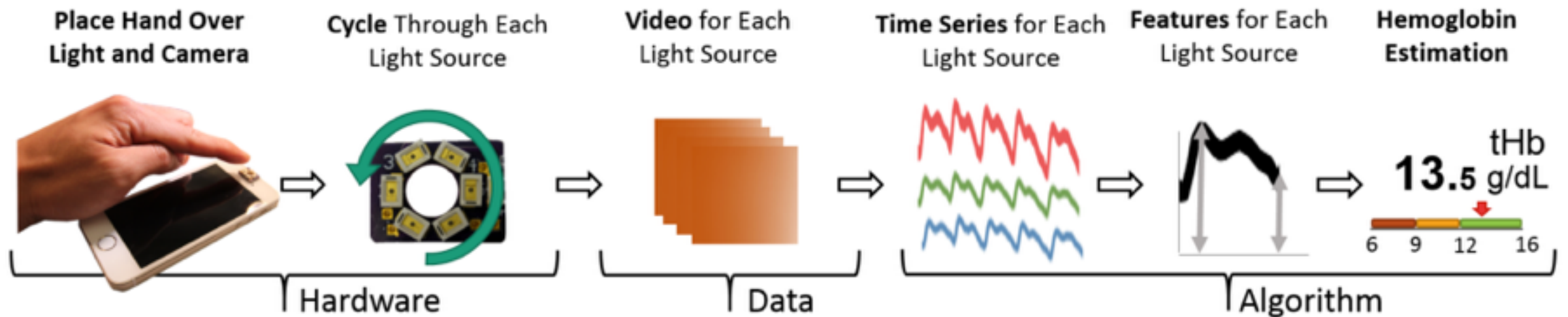


HemaApp



$$I_{measured} = I_0 e^{-\alpha[C]d}$$

HemaApp



$I_{peak} = I_{DC}$: The baseline intensity due to tissue.

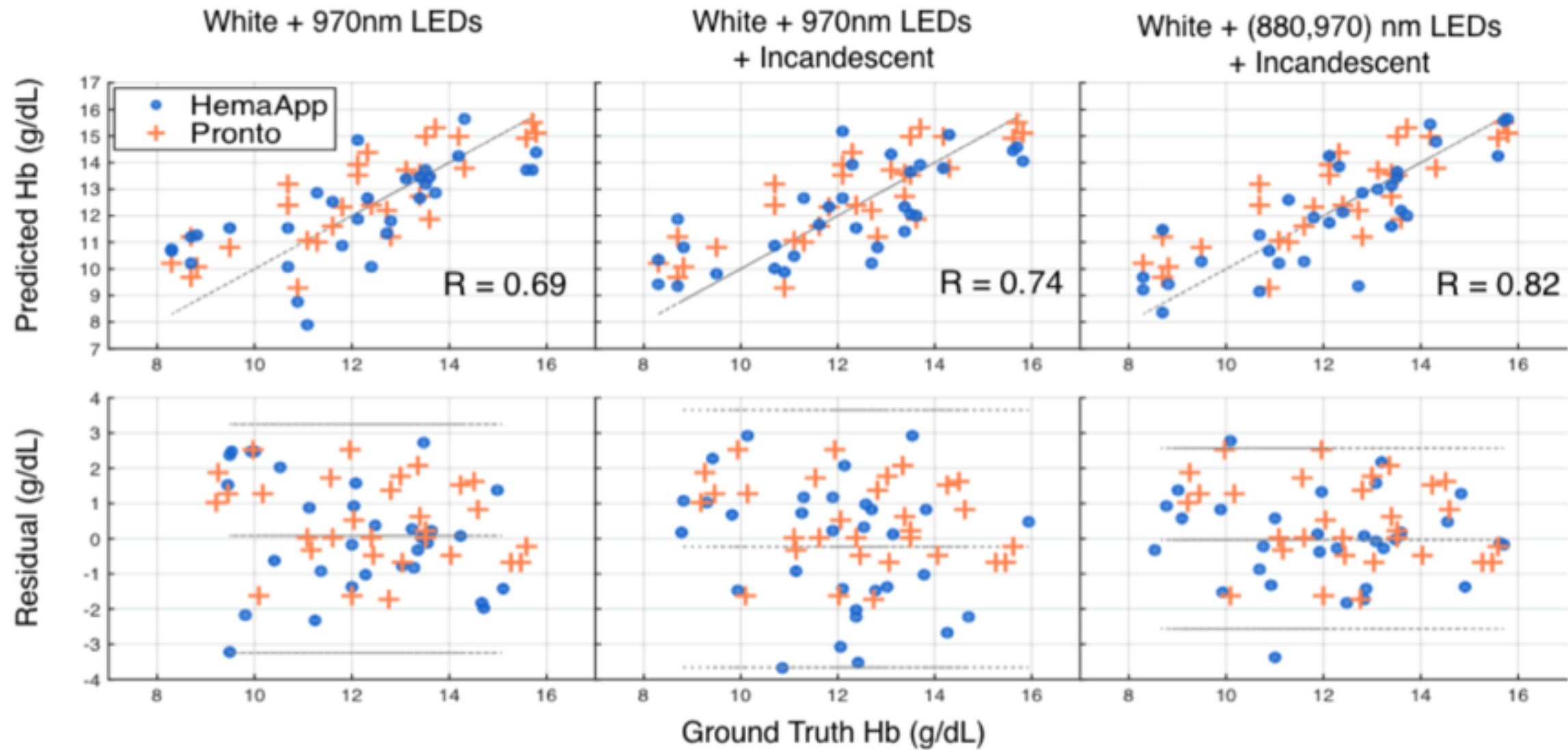
$I_{peak} - I_{trough} = I_{AC}$: The amplitude of the pulsatile absorption.

$\ln \left(\frac{I_{peak,\lambda}}{I_{trough,\lambda}} \right) = I_R$: The adjusted amplitude of absorption that eliminates the effect of tissue.

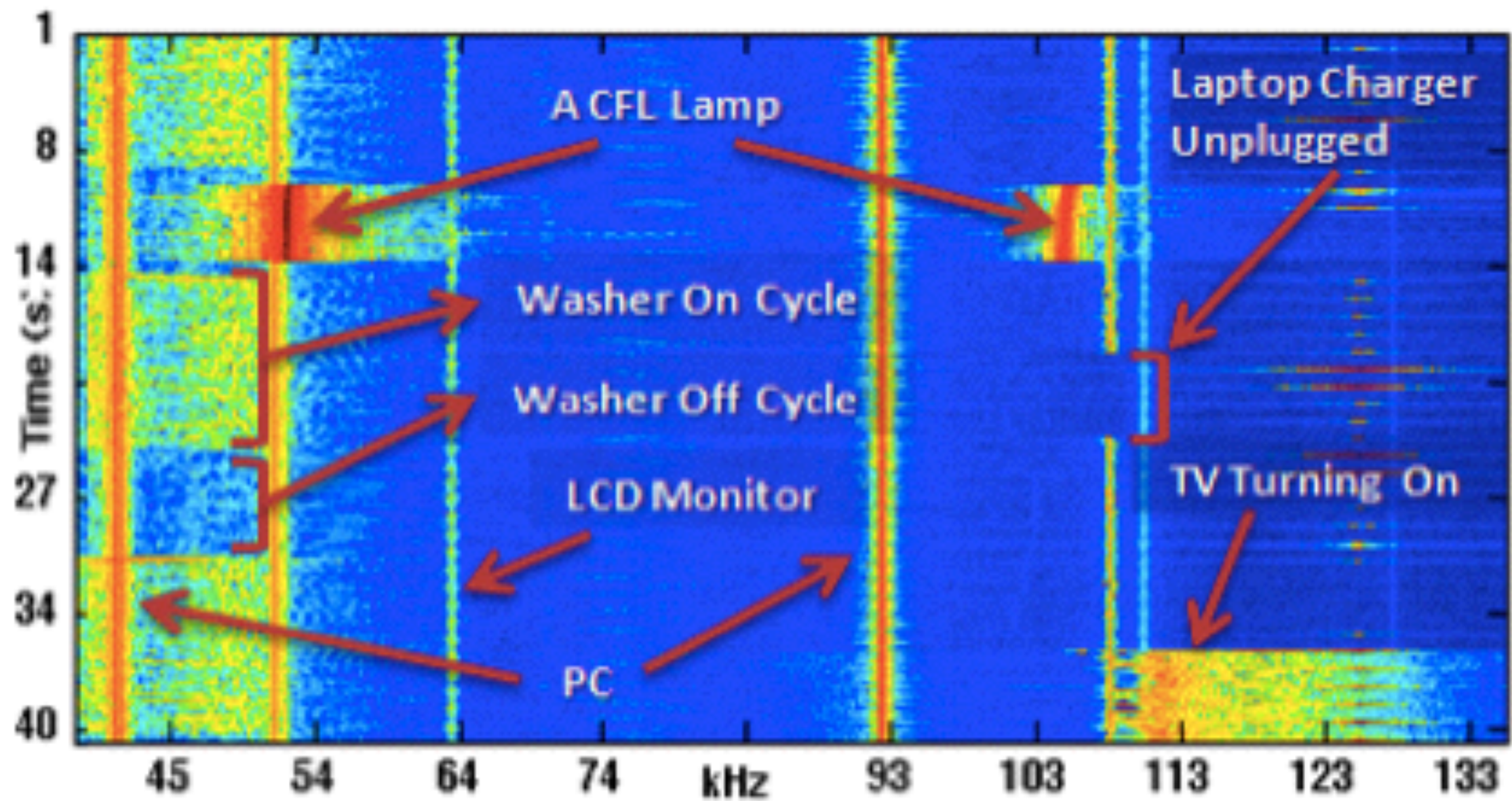
$I_{R,AC}(\lambda_i, \lambda_k) = I_{AC,\lambda i} / I_{AC,\lambda k}$: A pairwise ratio of pulsatile absorptions between all wavelengths.

$I_{R,ACDC}(\lambda_i, \lambda_k) = \left| \frac{(I_{R,\lambda i} - I_{R,\lambda k})}{I_{DC,\lambda i} - I_{DC,\lambda k}} \right|$: Absorption difference across wavelengths, adjusted with baseline.

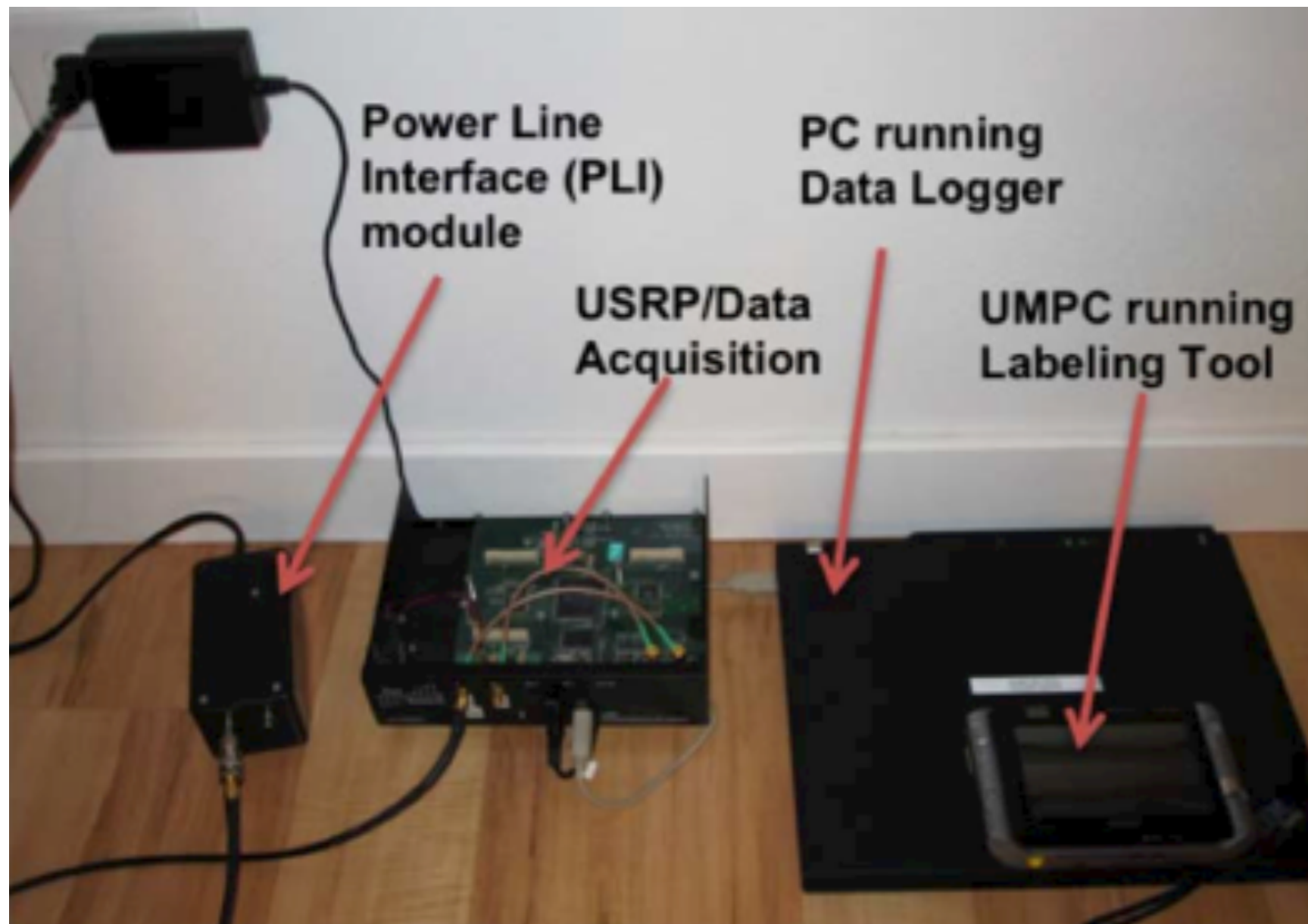
HemaApp



ElectriSense



ElectriSense



ElectriSense

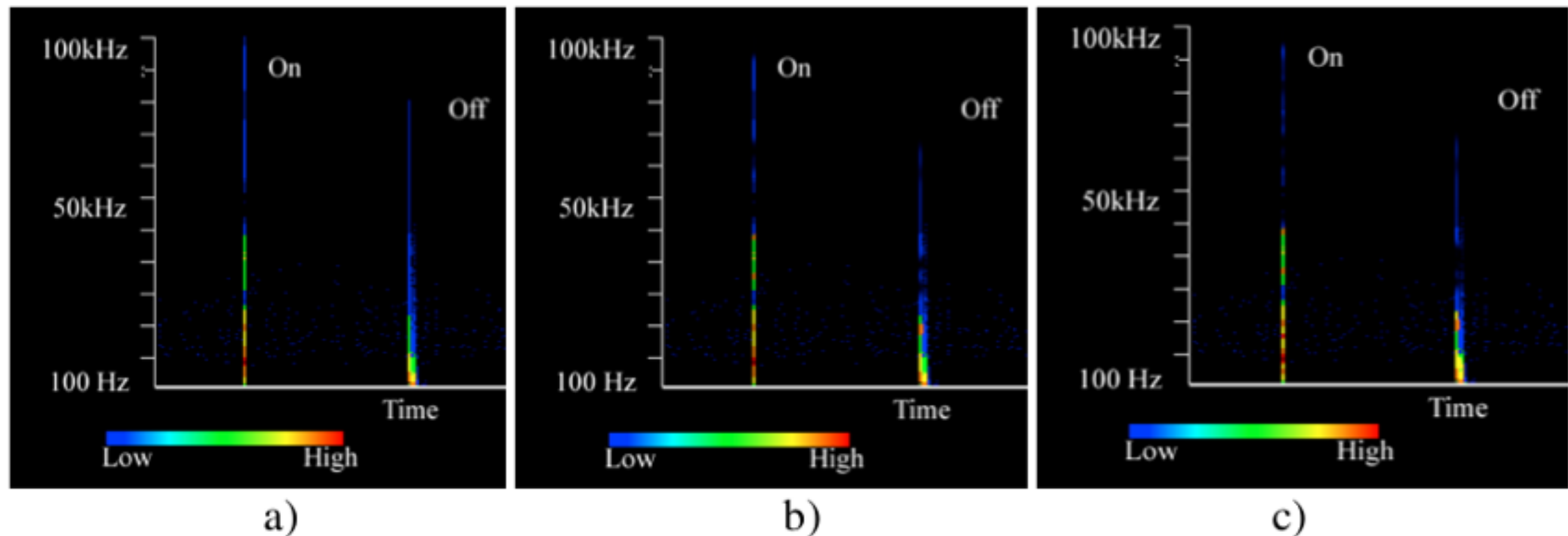
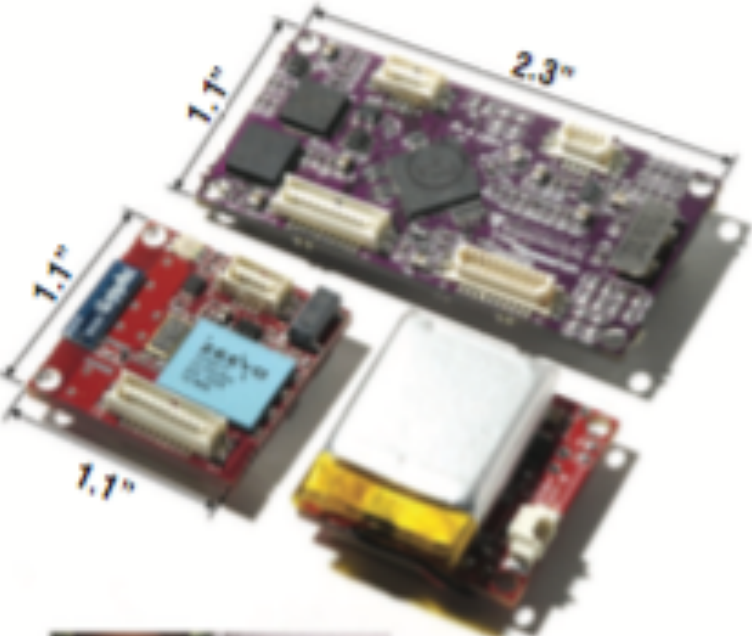


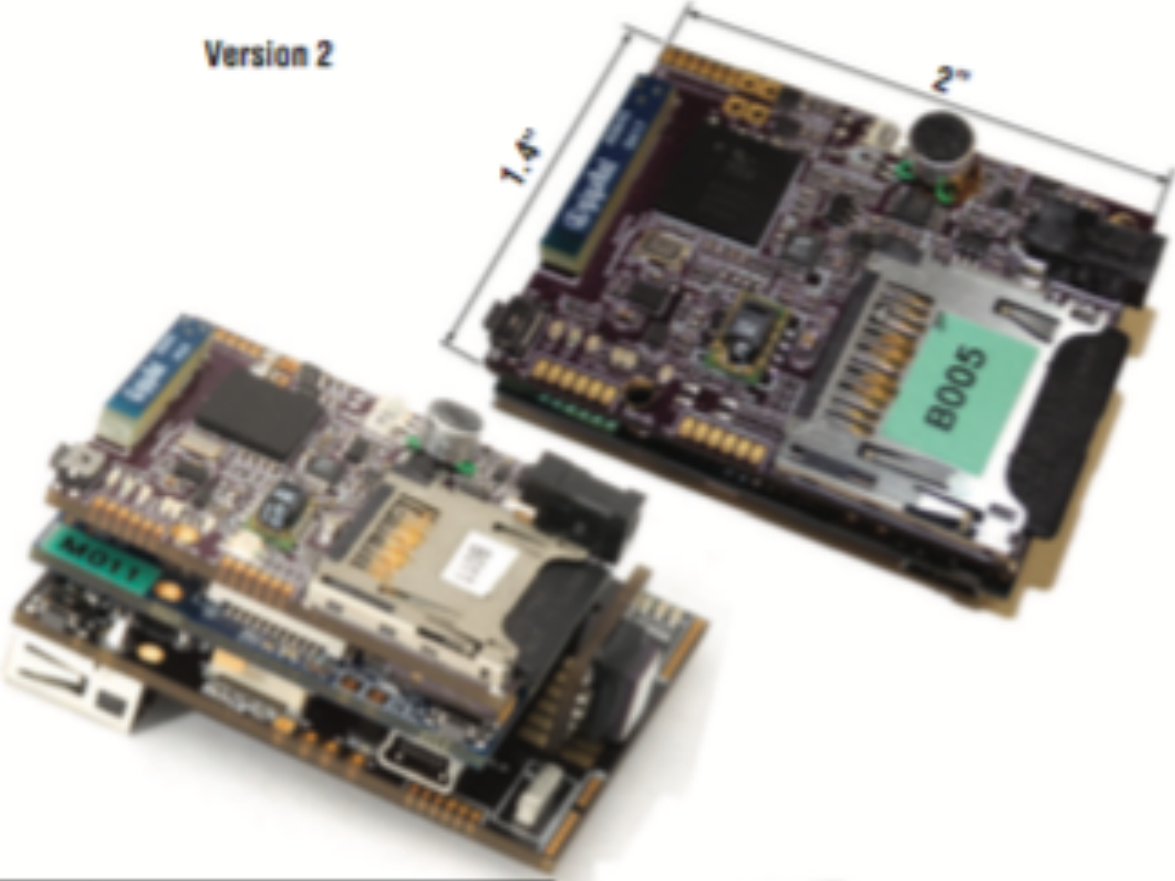
Fig. 2. Frequency spectrum of a particular light switch being toggled (on and off events). The graphs indicate amplitudes at each frequency level. Events in (b) were captured two days after (a), and events in (c) were captured one week after (a). Each sample is rich in a broad range of frequencies. On and off events are each different enough to be distinguished. In addition, the individual on and off events are similar enough over time to be recognized later.


MSP

Version 1




Version 2





Sensor description	Maximum sampling rate used
Electret microphone	16,384 kHz
Visible light photransistor	550 Hz
3-axis digital accelerometer	550 Hz
Digital barometer temperature	30 Hz
Digital IR and visible+IR light	5 Hz
Digital humidity/temperature	2 Hz
Digital Compass	30 Hz



Additional sensors on the location board: 3D magnetometers, 3D gyros, and 3D compass

Thursday

Please bring your laptop to class