I. Introduction

A. Beam Position Monitor (BPM)

A Beam Position Monitor (BPM) detects the centroid position of a beam non-destructively. Our in-flange type BPM hardware has four pickup electrodes—up, down, left and right—and everywhere else grounded. When a beam passes by, an image current of the same magnitude but opposite polarity is induced on the wall. The strength of the signal picked up by an electrode depends on its distance away from the beam. Therefore, by developing the processing electronics to analyze the pickup signals, one can deduced the beam position at a corresponding earlier time.

FIG. 1. Hardware for the BPM prototype

B. AWA Witness Beamline

This experiment is done solely on the witness beamline of the Argonne Wakefield Accelerator facility (AWA), operating in the single bunch mode with a typical charge volume of 1 nC, an energy of about 13 MeV and a beam current about 10 kA. The operating principle of the witness beamline is the following: the laser beam from the laser injector is reflected off a mirror to strike the semiconductor photocathode of the RF-gun, emitting electrons which are then initially accelerated and focused by the magnetic solenoid to about 8 MeV. After passing through the linear accelerating cavity (linac), the beam possesses an energy of about 13 MeV, and is further focused by three quadrupoles. The diagnostic system of the witness beamline so far only consists of Yttrium Aluminum Garnet (YAG) fluorescent screens, a destructive method of detecting the positions of each beam particle. Our prototype BPM system will be inserted after all of the quadrupoles.

FIG. 2. Labeled AWA witness beamline photo. Credit to John Power.

* Email me at chuanyin@uchicago.edu
II. MOTIVATION

The design of a BPM system should adapt to the functionalities and constraints of an accelerator. In other words, it is difficult and discouraged to simply apply the BPM designs used for first-class scientific experiments or huge commercial light sources to small accelerators. Too often, those BPM systems for big accelerators contain customized hardware and specialized processing electronics, sometimes developed and maintained by a special group of experts. This results in a higher fabrication cost and a barrier to creating electronics standards, both of which are disadvantageous to obtaining greater market share in the small accelerator economy, with main players being the R&D linacs and university accelerators on the research side, as well as the medical and industrial accelerators on the business side. For the small accelerators, strict constraints are placed on the compactness and cost-efficiency for each component. In light of this, Argonne Wakefield Accelerator facility (AWA) aims to prototype a cheap, compact and universal BPM system via mass-produced hardware, fast integrated circuits, and noise-protected Wi-Fi communication.

III. CURRENT PROGRESS

So far, the BPM system at AWA includes a button-type hardware installed on the beamline with four pickup wires, but without further processing electronics. AWA has designed and fabricated a new in-flange button-type BPM hardware soon replacing the current one, with a cost of $1000, at least ten times cheaper than an otherwise customized hardware thanks to mass-production. More tests and calibration on the new hardware are pending. For developing the processing electronics system, we performed a few sanity tests on the already-installed BPM hardware, to verify that it behaves the way as expected. For an ideal BPM hardware, the signal pulses should have linear dependence on both the beam positions and charge.

A. Beam-Position Dependence of the Signal Pulse

Qualitatively, the electrode that is the closest to the beam centroid position picks up the largest signal amplitude, and the farthest one picks up the smallest. In our testing setup, we adjusted the upstream horizontal and vertical correctors to steer the beam, observed the beam position on a downstream YAG screen, and recorded the scope traces for the four BPM pickup electrodes. While keeping the beam fully inside the inner rim of the YAG screen, we evenly divided the distance across and took 13 data points on the horizontal axis and 13 on the vertical one (of course, the origin was an overlapping point). We recorded the positions in pixels then converted them to centimeters using a calibration scale. For each beam position, we take 50 sets of scope traces to understand the extent of jitter in beam positions. For example, FIG. 3 shows one of the 50 traces when the beam is steered to the very left position. As expected, the left electrode picks up the largest signal, the right electrode the smallest, and the other two relatively the same.

Since the main source of error to the signal strength and the beam size is the fluctuation in the laser intensity, we employed a charge cut between two standard deviations of the charge scanned from the upstream ICT. A typical charge signal from the ICT is displayed in FIG. 4. Then, the average amplitude for each of the four pickup signals after the charge cut is plotted against each of the beam positions, as shown in FIGs. 5 and 6.

FIG. 3. The scope traces from the four pickup electrodes and the YAG screen image (center) with beam on the very left.

FIG. 4. A typical charge pulse.
The reason why each of the trends is not very linear with the beam positions might be due to the beam not centered longitudinally. Therefore, the signal amplitude depends on the angle between the passing beam and an electrode [1]. To eliminate this confounding variable, we take the difference in the signal amplitudes of the electrodes on the opposite ends, and plot that against the horizontal and vertical beam positions, in FIGs 7 and 8. As expected, we observe a highly linear trend in the direction that the beam is steered, and a rather flat trend for the other direction.

Furthermore, a more accurate way of determining the beam positions would be to use image processing software such as Matlab or Python to calculate the average center position weighted by intensity, assuming that the number of particles hitting the YAG screen on a pixel is proportional to the intensity of the pixel. Such study will also be important in determining the size of error in the beam positions.

B. Charge Dependence of the Signal Pulse

The amplitude of the all four signals summed together should also linearly depend on the beam charge. To test this, we performed a charge scan by applying five filters to the injected laser, resulting in 78%, 63%, 32%, 16%, and less than 1% transparency. A combiner (FIG. 9) is
inserted into the circuit taking the four BPM electrodes as its inputs, and it outputs to the scope as shown in FIG. 14. For each filter setup, we took 50 sets of signal data and recorded the ICT waveform. The charge is calculated by integrating the corresponding ICT pulse, up to a unit conversion factor and a DC background subtraction. We plotted the peak of the combiner output against the integrated charge in FIG. 11. Due to inherent fluctuation of the laser intensity, the charges produced by five distinct filter settings become somewhat continuous. Nevertheless, the linearity between the summed signal amplitude and the beam charge is confirmed with a $r^2$ squared value of 0.98.

![FIG. 9. A photo of the combiner.](image)

![FIG. 10. An example of the combiner output.](image)

![FIG. 11. Combiner signal amplitude as a function of charge.](image)

In summary, the current BPM hardware satisfies the general requirements that the differences in the signal amplitude of the opposite electrodes is linear with the beam position in that direction, and that the sum of all four signal amplitude is linear with the beam charge.

IV. WORKFLOW OF THE BPM PROTOTYPE

Although the pickup signals from the BPM hardware behave well, they are about 0.4 ns long that can only be seen by a scope with a sampling rate greater than at least 2 GHz. For one, it is expensive and inconvenient for small facilities to obtain fast oscilloscopes. On the other hand, displaying analog waveform in a commercial daily basis is an overkill since the only information needed from the waveform is its peak value, let alone that analog circuits are susceptible to noise. Hence, we decide to employ clever processing electronics to digitize the signal amplitude. However, it is a challenging task to keep the price of the design low. For a reasonably priced analog-to-digital converter (ADC), such as our ADS7951 chip, its sampling rate is only 1 MHz. Furthermore, since the ADS7951 is mounted on the Intel-Edison board, the speed is also limited by any constraints from the interface, resulting in an actual sampling rate of merely on the order of kHz, which obviously cannot see the 0.4 ns signal. Therefore, in order to pass the signal amplitude information to further electronics, either the sampling rate needs to be multiplied by 7 orders of magnitudes, or the short pulse needs to be elongated by 7 orders of magnitudes. Because the former approach requires much higher budgets, as experienced by larger accelerator facilities, we resort to the latter approach, by including basic electronics component such as four envelope detectors, four one-shot multi-vibrators, and four sample-and-hold chips.
A. Envelope Detector

The ADL5511 chip, displayed in FIG. 13, outputs a voltage that is proportional to the envelope of the input signal, and can operate from dc to 6 GHz on signals with envelope bandwidths up to 130 MHz, suitable for our usage on the 0.4 ns pulse.

FIG. 12. Workflow of the BPM system.

FIG. 13. A photo of the envelope detector chip.

As shown in FIGs 14 and 15, the envelope detector elongates a 0.4 ns-long BPM signal to an exponentially decay analog pulse with a sharp rising edge. The length of the output pulse is about 400 ns, and supposedly the amplitudes of the input and the output are proportional. More direct testing of the proportional relationship is needed, but if so, we would be able to achieve a three-orders-of-magnitude improvement on the duty cycle. The rest of the signal elongation is done by a sample-and-hold circuit.


FIG. 15. A signal after the envelope detector.
B. Sample-and-Hold Amplifier

A sample-and-hold amplifier (SHA), or sample-and-hold for short, is a common companion to an ADC, which triggers at one point of time of the input pulse and holds the signal voltage for longer. As demonstrated in FIG. 16, the SHA takes an analog 400 ns signal pulse and digital 100 ms pulse as an enable (EN). When the enable pulse is LOW, the SHA chip follows the signal; when the enable is HIGH, the SHA chip holds the instantaneous voltage multiplied by the amplification gain for the time the signal stays HIGH, i.e. 100 ms in our case. We decided upon a holding time of 100 ms due to the following constraints. The ADC later in the circuit has a sampling frequency of a few kHz, so in order for our pulse to be stably visible, the width of the SHA enable pulse has to be longer than 1 ms–and the longer the duty cycle the better. On the other end, the ADC takes only serial input and our accelerated bunch operates with a 500 ms period, so to allow enough time for us to combine all four analog SHA outputs into one and feed it to the ADC within one cycle, the width of the SHA enable pulse needs to be shorter than at least 125 ms. We also decided on triggering on the downward side of the signal because it would have less voltage error given the same time jitter.

We chose AD783 as our unity gain SHA device with a typical acquisition time of 250 ns. A photo of the chip is displayed in FIG. 17. The chip has an internal hold capacitor, and retains the held value with a droop rate of 0.02 µV/µs. Its aperture jitter is said to be 50 ps maximum, making it excellent for us to trigger accurately.

The connection diagram for the chip is displayed in FIG. 18. With that, we soldered four chips (for all four channels) to a breadboard as displayed in FIG. 19 and 20. A positive 5V dc power supply is connected to $V_{CC}$, a negative 5V dc power supply is connected to $V_{EE}$. The IN pin takes the input from the envelope detector, and the OUT pin is connected to the ADC module mounted on the Edison board.

![FIG. 17. A photo of AD783 chip.](image1)

![FIG. 18. Connection diagram for the AD783 chip.](image2)

![FIG. 16. A function diagram showing how the SHA circuit should work on the output of the envelope detector.](image3)

![FIG. 19. The front side of the 4-channel S/HA board.](image4)
In terms of the SHA enable pulse, clearly one desires an delay-adjustable, low-jitter digital pulse. At first, we attempted to use the Intel-Edison board to generate the digital pulse. A picture of the Edison board, which requires a 12 V DC power supply, is shown in FIG. 21. We programmed in the Arduino IDE software and compiled it to the Windows port of the Edison board, i.e. the one closer to the power supply.

However, the problem with using a mini pc like the Edison board to generate a fast digital pulse is a long delay time and a huge jitter in time. We performed a bench test on the sample-and-hold module to determine what is the shortest possible signal width that the system can hold, with two function generators: one producing a 2 Hz 1-ms-long square pulse to emulate the trigger signal from the control room, the other producing a width-adjustable sub-microsecond pulse with a known delay from the first one to emulate the BPM signal pulse after the envelope detector. The sample-and-hold functionality is achieved when the rising edge of the Edison digital pulse lies within the short signal pulse, i.e. the delay between the two function generators matches with the digital pulse from Edison. As it turned out, the sample-and-hold module is not able to stably hold any signal narrower than 2.5 $\mu$s, or any delayed by less than 80 $\mu$s from the trigger pulse. Both problems are due to the shortcomings of the digital pulse from Edison. In particular, the long delay time might amount to Edison tackling other tasks first; and the high jitter might be because Edison’s CPU sending an interrupt signal whenever a task arrives, which becomes especially problematic when it samples fast. As a remedy, the fast digital pulse is instead generated by a hardware chip, as will be introduced below in the Dual One-shot Multivibrator section.
C. Dual One-shot Multivibrator

Our choice of the delayed digital pulse generator is a 16-lead highly stable dual non-retriggerable one-shot chip DM74LS221N, or one-shot for short. Its output pulse width ranges from 30 ns to 70 s, suitable for our usage of generating a 100 ms long digital pulse. We soldered the chip to a breadboard as shown in FIGs. 26 and 27 according to the connection diagram (FIG. 24) and the function diagram (FIG. 25).

![FIG. 23. A photo of the DM74LS221 chip.](image)

![FIG. 24. The connection diagram for the DM74LS221 chip.](image)

![FIG. 25. The function diagram for the DM74LS221 chip.](image)

The usage of each pin is summarized in TABLE I. A 5V dc power supply is used as both Vcc and digital HIGH. By setting A1 to LOW, CLR1 to HIGH, and B1 to the trigger IN, at the rising edge of the input, the first RC circuit outputs a square pulse with an adjustable width as Q1 and an inverted pulse with an adjustable width as $\overline{Q1}$. The length of the first output signal is simply controlled by the time constant of the first RC circuit, resistor R1 and capacitor C1. Then, we feed this $\overline{Q1}$ output to pin 10, i.e. B2. By also setting A2 to LOW, CLR2 to HIGH, the second RC circuit outputs a square pulse with an adjustable width as Q2 and an inverted pulse with an adjustable width as $\overline{Q2}$. The length of the second output signal is simply controlled by the time constant of the second RC circuit, R2 and C2. Hence, in the context of us producing a delayed digital signal, the first RC circuit determines its delay (with respect to the trigger pulse), and the second RC circuit determines its width. More specifically, to aim for a $10 \mu s$ delay and a 100 ms long pulse, we fixed C1 to be 1.40 nF and C2 to be 2.90 $\mu$F, and adjusted the resistances of two variable resistors to be about 10 kΩ for R1 and 50 kΩ for R2.

<table>
<thead>
<tr>
<th>Pin</th>
<th>General</th>
<th>Customized</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A1</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>B1</td>
<td>Trigger IN</td>
</tr>
<tr>
<td>3</td>
<td>CLR1</td>
<td>Vcc</td>
</tr>
<tr>
<td>4</td>
<td>Q1</td>
<td>$\overline{Q1}$</td>
</tr>
<tr>
<td>5</td>
<td>Q2</td>
<td>NC</td>
</tr>
<tr>
<td>6</td>
<td>$\overline{Cext1}$</td>
<td>R2-C2-Vcc</td>
</tr>
<tr>
<td>7</td>
<td>$\overline{Rext/Cext2}$</td>
<td>R2-Vcc</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>A2</td>
<td>GND</td>
</tr>
<tr>
<td>10</td>
<td>B2</td>
<td>Q1</td>
</tr>
<tr>
<td>11</td>
<td>CLR2</td>
<td>Vcc</td>
</tr>
<tr>
<td>12</td>
<td>Q2</td>
<td>OUT</td>
</tr>
<tr>
<td>13</td>
<td>Q1</td>
<td>NC</td>
</tr>
<tr>
<td>14</td>
<td>$\overline{Cext1}$</td>
<td>R1-C1-Vcc</td>
</tr>
<tr>
<td>15</td>
<td>$\overline{Rext/Cext1}$</td>
<td>R1-Vcc</td>
</tr>
<tr>
<td>16</td>
<td>Vcc</td>
<td>Vcc</td>
</tr>
</tbody>
</table>

TABLE I. Summary of pinout for our DM74LS221N chip.
Overall, according to our cold test, the one-shot chip together with the sample-and-hold circuit can stably hold any signal such that its width is greater than 70 ns, and it is delayed from the trigger pulse by more than 600 ns. Please refer to FIG. 28 for the lower limits, namely the shortest signal width and closest signal delay, for the SHA and one-shot system. The fast speed of the circuit permits its usage on elongating the 100 ns pulse outputted from the envelope detector, supposedly. Moreover, no jitter above 1 ns was observed in the one-shot output.

<table>
<thead>
<tr>
<th>One-shot characteristics</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Delay</td>
<td>600 ns</td>
<td>17.5 µs</td>
</tr>
<tr>
<td>Output Width</td>
<td>1 ms</td>
<td>∞ (&gt;450 ms)</td>
</tr>
</tbody>
</table>

TABLE II. A summary for SHA using the DM74LS221N chip.

FIG. 26. The front side of the one-shot board.

FIG. 27. The back side of the one-shot board.

FIG. 28. The lower limit in the signal pulse width that the SHA system can hold.

FIG. 29. A zoomed in view of the SHA circuit holding a 400 ns pulse.
FIG. 30. A zoomed out view of the SHA circuit holding a 400 ns pulse.

FIGs. 29 and 30 display different views for holding a 400 ns pulse, emulating the elongation of the BPM signals from the envelope detector. While the emulation can be exact in the timescale, their pulse shapes are different, so a better cold test can be conducted. As a counterexample, FIG. 31 represents the scenario such that the SHA doesn’t hold.

Previously, Edison was only able to generate a digital pulse whose signal width is greater than 2.5 $\mu$s and signal delay is greater than 85 $\mu$s. And its output delay has a significant jitter of about 5 $\mu$s. Thus, we have made an about 1000 fold improvement, thanks to replacing a CPU device with a hardware piece.

V. CONCLUSION

Overall, we devised a method to integrate multiple cheap components to process the four original BPM pickup signals, which involves an envelope detector, a peak detection and digitization system, and Wi-Fi sendout. The project conducted a beam test that confirms the linear relationship between the raw signal with respect to the beam charge and beam position. As for the bench test on the envelope detector, it behaves linearly with charge, but is not sensitive enough to position. Currently, we are examining whether the signal is inundated by noise, and striving to increase the signal-to-noise ratio by a band-pass or a low-pass filter. The cheap peak detection system was separately built and cold-tested. Given that the envelope detector works well, i.e. stretching the original signal to above 400 ns long while having its amplitude proportional to the raw signal, the following peak detection system should cooperate, because it can hold any signal longer than 70 ns in a test with functional generators emulating the beam signals. Furthermore, in the cold test, the analog output from the SHA chip is long enough for the analog-to-digital converter. However, more beam tests and integrated tests need to be done on the peak detector.

In terms of the Wi-Fi sendout module, we have managed to use a router to connect a computer to the Edison board (also plugged into it) wirelessly. The next step is to send simple signals between the Edison board and a remote computer.


Listing 1. Arduino code for Elongating a Digital Signal through Edison

```c
/*
  Digital 2 Hz, 25 ms input (act as trigger)
  Digital 2 Hz, 100 ms output (act as S/H input)

Read the 25 ms square pulse from function generator to digital input
Print the value to console 0−1023 to the serial monitor
When the pulse arrives, delay for 0.2 ms, then produce digital output that lasts for 100 ms
Print digital output to serial monitor too

The circuit:
* Function generator connected to digital I/O pin 0 and ground as input.
* Scope connected to digital pin 1 and ground as output.

created 21 Jul. 2017
*/

int outPin = 3;
int inPin = 2;
int val = 0;   // variable to store the read value

void setup()
{
  Serial.begin(200000);
  pinMode(outPin, OUTPUT);   // sets the digital pin 3 as output
  pinMode(inPin, INPUT);     // sets the digital pin 2 as input
}

void loop()
{
  if (digitalRead(inPin) == 1)
  {
    delayMicroseconds(200);
    digitalWrite(outPin, LOW);
    delayMicroseconds(100000);
    digitalWrite(outPin, HIGH);
    delayMicroseconds(50000);
  }
  else
  {
    digitalWrite(outPin, HIGH);
  }
  val = digitalRead(outPin);
  Serial.print(val);
}
```
Listing 2. UDP Wi-Fi Setup by Edison (credit to Jiahang Shao).

```c
#include <SPI.h>
#include <WiFi.h>
#include <WiFiUDP.h>

// declarations
int inPin = 2;  // variable to store the read value
int status = WL_IDLE_STATUS;
char ssid[] = "Wiboard";  // your network SSID (name)
char pass[] = "123456789";  // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;  // your network key Index number (needed only for WEP)
unsigned int LocalPort = 22222;  // local port to listen on
unsigned int Port = 12344;  // local port to listen on
IPAddress remoteIp(192,168,253,1);
char SendBuffer[15];
unsigned long measuretime = 0;
int measuremax = 0;
int measuretemp = 0;
int count = 0;
int k_temp = 0;

WiFiUDP Udp;

// setup
void setup()
{
  Serial.begin(200000);
pinMode(inPin, INPUT);  // sets the digital pin 7 as input
  // Initialize serial and wait for port to open:
  Serial.begin(115200);
  // Serial.begin(9600);
  while (!Serial) {
    ;  // wait for serial port to connect.
  }

  pinMode(A0, INPUT_PULLUP);

  // check for the presence of the shield:
  if (WiFi.status() == WL_NO_SHIELD) {
    Serial.println("WiFi_shield_not_present");
    // don't continue:
    while (true);
  }

  // attempt to connect to WiFi network:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to \"SSID: \"");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:
    delay(10000);
  }
  Serial.println("Connected to wifi");
  printWifiStatus();
  Serial.println("\nStarting connection to server ...");
  // if you get a connection, report back via serial:
  Udp.begin(LocalPort);
}
```
```cpp
void loop(){  
  val = digitalRead(inPin);  
  Serial.println(val);  
  Udp.write(val);  
}

void printWiFiStatus() {  
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");  
  Serial.println(WiFi.SSID ());  

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP ();  
  Serial.print("IP Address: ");  
  Serial.println(ip);  

  // print the received signal strength:
  long rssi = WiFi.RSSI();  
  Serial.print("signal strength(RSSI): ");  
  Serial.print(rssi);  
  Serial.println(" dBm");  
}
```