Greendicator: Enabling Optical Pulse-Encoded Data Output from WSN for Display on Smartphones

Shin-Yi Chang*, Jo-Ping Li*, Hua-Min Tseng*, Pai H. Chou†*
*Department of Computer Science, National Tsing Hua University, Taiwan
†Department of Computer Science, University of California, Irvine, USA
{littlesakt, zeroping.tw, thm822, pai.chou}@gmail.com

***Abstract*—Greendicator is an indicator system that enables wireless embedded sensing systems to output data encoded as LED pulses to be decoded and viewed on smartphones. The transmitter encodes the message to be displayed in the form of modulated light pulses emitted with an existing visible-light LED or other light-emitting devices such as IR, laser diode, or light reflector. The receiver uses a camera-equipped smartphone to sense the light pulses and to decode the original message. Greendicator can be extended using the camera's built-in flash-light as an acknowledgement to enable success communication. To develop the real-time decoding system, we apply GPU on the smartphone to process the image frames. An API is provided for programmers to incorporate Greendicator in their systems by simple function calls. Experimental results show the Greendicator can be a valuable mechanism for supporting existing RF-based networks while occupying a small memory footprint.**

I. Introduction

Wireless sensor networks (WSNs) applications usually demand low cost or small size, thereby forcing the hardware to be built with minimal resources and to forego features such as text displays. The most common way for a wireless sensor node to output messages is to use its RF transceiver. The bandwidth of RF transceivers can be high, making them an excellent choice for applications to communicate with each other as well as for firmware updates. However, RF transceivers need to be properly configured to work in the first place. Protocol stacks that support automatic configuration exist, but their code size may be too large for resource-constrained platforms, which often require manual configuration. A converse problem is that when all RF modules are working, one might not be able to tell which node is which, because there is no way to "see" the wireless links, and any self-reported names by the nodes may not be very meaningful especially if they are all similar. Moreover, RF-based communication is intrinsically broadcast and is prone to eavesdropping and replay attacks. For all these considerations, we believe that another means of wireless communication to complement RF is needed.

To provide the complementary functionality to existing WSNs, we propose a communication approach called Greendicator, an indicator system for augmenting LED-equipped embedded systems with text-display capability. Greendicator provides a form of optical wireless communication (OWC) and offers several advantages over the RF, including the unregulated, potentially significantly higher available bandwidth without electromagnetic interference (EMI). The use of conventional components such as laser diodes enables one to easily achieve longer communication distance than RF in a small device consuming low power. The strong directionality also means OWC inherently provides a high degree of privacy and security against eavesdroppers in short-range line-of-sight links.

This paper is organized as follows. We survey previous work in related areas, followed by an overview of the proposed system. We explain the modulation scheme, communication protocol, image processing, and extended bi-directional communication. Experimental results show Greendicator to be particularly applicable to embedded systems in terms of the versatility in multi-optical sources and the decoding accuracy over distances and angles, especially considering the very limited resource on the sensor nodes.

II. Background and Related Work

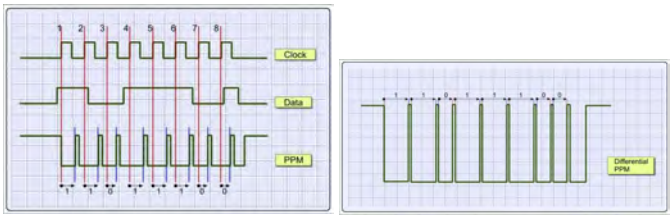
This section first provides a background on Greendicator. Then, we discuss related work on optical information indicators and optical wireless communication.

A. Background

1) Optical Wireless Communication: In recent years, OWC has become a viable and promising communication modality that supplements existing radio-based wireless technologies. OWC offers numerous advantages over RF. For example, radio signals suffer from electromagnetic interference (EMI), but OWC has huge, unregulated, and available bandwidth without EMI. Moreover, OWC signals can travel a very long distance when focused. Such properties become especially important in operating environments such as airplanes, where radio systems can interference with the on-board navigation instruments. Optical signals are absorbed by dark objects, diffusely reflected by rough surface, directionally reflected from shiny surface, and able to penetrate glass but not walls or opaque barriers. The signal confinement inherently enables a high degree of privacy and security against eavesdroppers for OWCs.

2) Modulation: Modulation is a key component of telecommunication, especially OWCs, because a proper modulation scheme makes the signals stand above the noise floor. We introduce the modulation schemes used in Greendicator.

Pulse-Position Modulation (PPM) is a well-known orthogonal modulation technique widely used in optical communication systems [7]. Data are encoded as short pulses that have the amplitude, where logical '0's and logical '1's are distinguished by the delay of the pulse relative to the start of the bit cycle. An example is shown in Fig. 1a. In other words, a short delay represents a logical '0', and a longer delay represents a logical



(a) (b)

Fig. 1: (a) PPM transmission example, (b) DPPM transmission example.

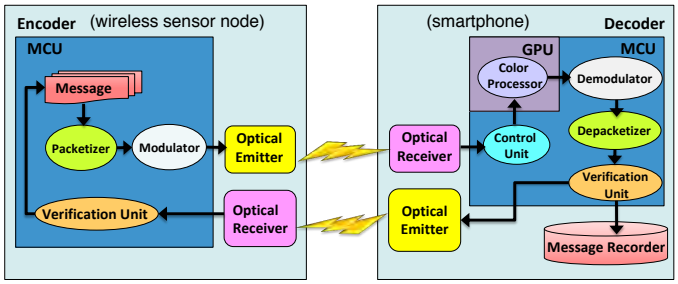


Fig. 2: System overview.

'1'. The delay is not standardized and can be arbitrarily defined to meet the system requirements.

However, since PPM requires the receiver to synchronize its bit cycle with the sender to properly measure the pulse delay, it is inherently sensitive to multipath interference caused by a receiving signal containing one or more echoes of transmitted pulses at a time. Because PPM encodes information into the time of arrival, the multipath effect makes it extremely difficult to accurately determine the correct pulse position corresponding to the transmitted pulse.

Differential Pulse-Position Modulation (DPPM) [8] is proposed to eliminate clock synchronization. As shown in Fig. 1b, DPPM encodes each data bit relative to the previous pulse only, rather than the start of the sender's bit cycle, so the receiver needs to measure only the differences in the arrival time of successive pulses. This modification makes it possible for DPPM to limit the propagation errors to adjacent symbols, and an error in measuring the differential delay of one pulse will affect at most two symbols, instead of affecting all successive measurements as in PPM. One characteristic about DPPM is its data-dependent timing: the more zeros there are in the data, the less time it needs for transmission.

High energy efficiency and high utilization of bandwidth are attractive features of DPPM, and we design our optical communication system in a directed link where beams proceed in a line-of-sight (LOS) path without multipath propagations. Therefore, we use DPPM instead of other modulation schemes for Greendicator.

B. Related Work

OWCs have received much attention and grown tremendously in the last decade. We list several related works on OWC with cameras as the sensing device. Arai [5] focuses on parallel OWC systems using a matrix of LEDs as the transmitter and a high-speed camera as the receiver for road-to-vehicle communication. The hierarchical encoding scheme, which allocates data to spatial frequency components depending on the priority, makes it possible to receive the high-priority data even if the receiver is too far from the transmitter to receive the entire data. The data transmission throughput is higher than that of a single LED due to the array structure. Their data decoding is processed off-line, and powerful hardware support is required.

Bokode [3] is a type of tag with thousands of times higher information density than a barcode. It is much smaller than a barcode and is circular in shape with a diameter of 3 mm. A bokode consists of an LED covered with a mask and a lens. It is readable from different angles and from 4 meters away by an SLR camera. Currently, bokode costs about five dollars to produce as the LED has to be combine with a specific mask and lens, and the decoder now is implemented with an SLR camera.

FlashLight [2] uses a mobile phone and an interactive tabletop to achieve bi-directional communication. It uses a color-based encoding method to transmit data. The mobile phone is placed on the tabletop, and the color on the tabletop changes continuously to transmit binary data. The flashlight equipped on a smartphone can be used as the feedback source to achieve bi-directional communication, which is similar to our Greendicator system. Since they use the transition between different colors as the encoding source, specific light emitted device such as color-changeable LED is required and thus is not as simple as our system.

III. System Overview

Greendicator is organized as an encoder/decoder architecture through optical communication, as shown in Fig. 2. Encoding can be performed by an existing sensor platform with an optical emitter, while decoding can be executed on a conventional camera-equipped smartphone.

The encoder side starts up by parsing generated messages and recomposing them into the correct packet format. Then, the basic modulation unit transforms the packets by the chosen encryption type into a sequence of binary optical signals to be sent by the optical emitter. Due to the limited memory and processing capability on many WSN platforms, the encoder side is kept simple by concentrating only on sending messages as optical pulses.

On the decoder side, the smartphone's camera receives the optical pulses, extracts the message, and displays it either as overlay on the live video from the phone's camera or in a dialog history similar to instant messaging. The decoder side performs the core image-recognition algorithm by a color threshold continuously as part of the decoding process.

We take advantage of the GPU on the smartphone to enable the decoder to handle a higher frame rate from the camera, and therefore the encoder can shorten modulation periods to achieve a higher transmission rate. However, the higher data rate can lead to a higher bit error rate. To increase reliability, the encoder performs error-correctable encoding by inserting redundant bits into the original data.

(a) (b) (c)

Fig. 3: EcoSD ultra-compact wireless sensor platform (bottom, top) and the OP520 infrared phototransistor

Greendicator provides an API similar to C `<stdio.h>` for programmers to display the message. The usage is similar to the `printf` function, with support for several common data types, including constant string, char, and 16-bit integer. Unlike `stdio`, however, Greendicator API supports encoding schemes called *ASCII* and *Table*. ASCII encoding is what programmers expect by default, as all MCUs work with ASCII characters. Table encoding, on the other hand, is an attempt at conserving bandwidth by transmitting a message index rather than the text, and the programmer has to make only minimal adjustment to their code. For Table encoding, our tool automatically numbers each print function by assigning a message index and generates a string table file to be uploaded to a server. The smartphone can then connect to the server and download the string table file to decode the messages. When there is no network connection, however, the smartphone and sensor node will fall back to ASCII scheme.

IV. Encoder

This section describes an encoder design for a resource-constrained wireless sensing platform. Our initial target is an ultra-compact wireless sensor platform called EcoSD, as shown in Fig. 3, which contains an 8-bit, 8051-based micro-controller unit (MCU) with 16 KB program flash and 1 KB data SRAM, but it can be easily ported to any other embedded processors. One extension is to add a phototransistor, the OP520 (Fig. 3c) to receive acknowledgment.

A. Communication Protocol

1) Packet Format: The packet format is composed of the *preamble*, *payload*, and *postamble*. The preamble indicates the start of the packet, the payload contains the data to transmit, and the postamble marks the end of the packet, as depicted in Fig. 4. We convey the nodeID in a separate packet first. For the rest of the packets that contain the messages in API function, the packetizer constructs the packet with the aid of the API parser (Section IV-B1); the depacketizer deconstructs the packet by checking the received packet length and current FSM state. Fig. 5 depicts the FSM of the depacketizer.

The decoder starts from the *Init* state and waits until it receives the long preamble and makes a transition to the *received-header* state. The preamble can be detected when the count of consecutive frames that contain the detected target LED light exceeds the predefined threshold. After receiving the nodeID, the decoder sends back a light pulse to notify the node of conveying the packets that contain the data in the API function and waits for the ack from the node. Upon receiving the ack (three 00 pulses) from the node, the decoder starts the decoding process on the received messages. After

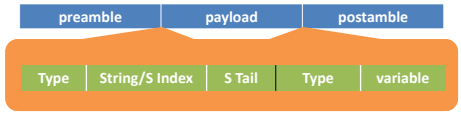


Fig. 4: Communication protocol and packet format

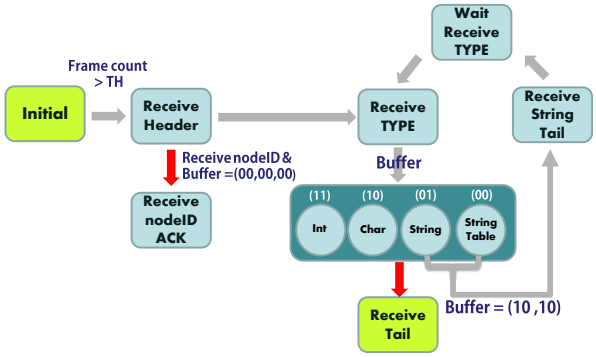


Fig. 5: FSM of depacketizer.

the preamble, the sender indicates the encoding format, and the decoder decodes data accordingly (*Receive Type* state in FSM).

To handle the parameter to be formatted, the protocol includes a state named *Receiving String Tail*, also called a sentinel. If the received string or string table is appended with a sentinel, then it means it is the end of the string and it switches to receiving the variable to be formatted. After receiving the postamble, the decoder ends the packet processing and displays the formatted result.

2) Modulation Scheme: The modulator and demodulator are the fundamental components in the encoding and decoding processes. The modulator converts the logical values in the packet into the sequence of timed pulses and spaces, whereas the demodulator restores them back into the data. We implement Greendicator encoding system on a resource-constrained sensor platform named EcoSD. The modulation scheme must be efficient but of low computational complexity. The DPPM scheme that we have discussed in Section II-A2 fits our requirements, since it encodes a signal regardless of a clock but takes its time reference to be the falling edge of the previous pulse.

Fig. 6 shows our modulation scheme. To increase the transmission rate, we encode a two-bit group in a modulated pulse-space transition, thereby doubling the transmission rate. The choice of the duration is based on the limit imposed by the camera's frame rate. Since the average frame rate in our system is around 20 fps, the decoder must process each frame within 50 ms. To ensure that the decoder can successfully detect the pulse, we define 100 ms as the shortest period in the modulation scheme by the Nyquist theorem [6]. To identify the preamble, we define a 500 ms pulse accompanied by a 100 ms space. Conversely, the postamble is composed of a 100 ms pulse accompanied by a 500 ms space. The long preamble and postamble can be easily distinguished during decoding.

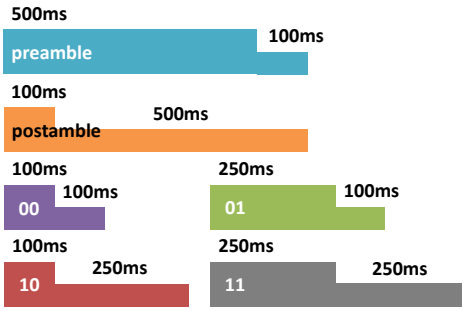


Fig. 6: The modulation scheme in Greendicator.

To transmit two bits at once, we map their permutations to four modulation patterns:

- *00* as a 100 ms pulse followed by a 100 ms space,
- *01* as a 250 ms pulse followed by a 100 ms space,
- *10* as a 100 ms pulse followed by a 250 ms space, and
- *11* as a 250 ms pulse followed by a 250 ms space.

In the demodulation process, the number of the pulses and spaces will be counted in each camera frame. We calculate the ratio between pulses and spaces to identify the patterns. Considering the variability of the camera frame rate, the range of each modulation pattern will be dynamically adjusted in each demodulation round. Since the defined periods (100 and 250 ms) are factors of that of the preamble (500 ms), we can infer the range of frame counts in each modulation pattern from the frame counts of the preamble.

B. Tools and Extensions

1) Node API Parser: In this section, we explain the parsing method behind the node API, which helps the packetizer to construct a packet. To provide a common interface, we define `_std_print` as the `printf` function in C. Fig. 7 illustrates how the API parser constructs the optical packet. When entering the print function, the parser first checks the output-encoding format such as ASCII and Table. Then, the packetizer can build the preamble accordingly and construct the rest of the packet in compliance with the input messages.

The function is composed of a string pointer to read the constant message and an integer to read the assigned variable. The format specifiers `%d` and `%c` cause the parser to pass the patterns of data types and variables into the packetizer. We use two bits to indicate four data types, including 00 for String Table, 01 for String, 10 for `char`, and 11 for `int` type. Likewise, with the variable identifier, the parser can know that a variable is to be appended after the string by putting a string tail to separate them. We use a flag to indicate the state as either "string only" or "string with a variable."

2) Error-Correcting Method: Greendicator implements Hamming code [4] for error correction. When using Hamming code, several parameters need to be checked, such as the *message length* $k = 2^r - r - 1$ and the *block length* $n = 2^r - 1$. When $r = 4$, the maximum message length is 11, so the total block length is 15. Since the full ASCII code can be represented in eight bits, we insert four redundant bits at positions 1, 2, 4, and 8 for error correction, and the full packet

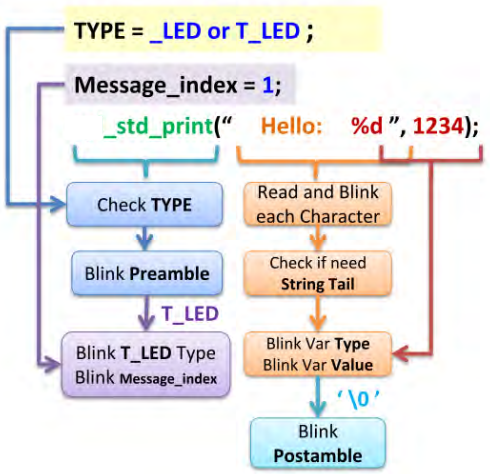


Fig. 7: The packet construction flow through API parser.

length for each character is 12. Such Hamming code enables detection of up to two error bits and correction of up to one error bit. The decoder side validates the received packet by adding up the bits in the checking sequence for each parity check.

3) Bi-directional Communication: The extension to bi-directional communication in Greendicator can be achieved through the cooperation between a timer and the OP520 phototransistor on EcoSD. We use OP520 to receive the light from the smartphone, and the timer periodically checks whether the received light is above the defined light-intensity threshold, although this can also be done in interrupt style using a voltage comparator. Although OP520 senses light in the near-infrared regime, it is more robust to the noise from the ambient light. The node side checks every *three* seconds and maintains a *nine*-second detecting window to receive the light pulses from the smartphone.

To configure the bi-directional functionality, the checking function counts the number of light pulses emitted from the smartphone and sets a flag upon reaching the target count. Then, the node changes its status and decides the encoding scheme. Initially, we make the node convey its nodeID repeatedly. Once the node detects a sequence of legal light pulses, it transmits back an ack to the phone and breaks the initial loop to enter another message-conveying loop. In the message-conveying step, the node transmits the message according to the encoding scheme defined by the flag. The message-conveying loop will terminate, and the node will go back to the initial state when receiving the reset signal from the phone.

V. Decoder

A. Mode-Transition Handler

We define three modes in bi-directional communication: *ASCII blinking*, *Table blinking*, and *Reset node*. The smartphone toggles the flashlight in different periods to represent each mode. Fig. 8 depicts the transition between different modes with different light patterns. Before the node starts a transmission, the smartphone checks if it has network connection. If so, it tells the sensor node to use Table encoding

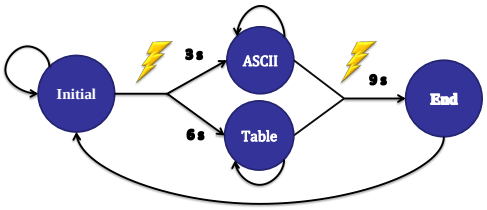


Fig. 8: Mode transition between different states.

by sending a 6-second pulse; otherwise, it tells the node to use ASCII encoding by sending a 3-second pulse. To end the decoding process, the user can touch the reset button on the screen and the flashlight will be toggled to send a 9-second pulse. After detecting the pulse, the node returns to the initial blinking nodeID state.

B. Color-Threshold Detection

Color-threshold detection is the core image-processing technique in the implementation of the decoder. With the aid of OpenGL ES, image processing is executed on the GPU, thereby speeding up the camera frame rate. We modify the color-threshold template [1] in the image processing part because of its simplicity and versatility. Any kind of light detectable by the camera can be an optical source for our system. Users define the target LED color by pointing at the lighted region on the screen and sliding their fingers to adjust the color threshold. The color threshold is used to define the similarity of the color between the target LED and every pixel in the frame, and user-defined parameters such as color, threshold, and the pointing coordinates will be passed and recorded in the OpenGL shader program. The transmission of parameters between the shader and the main program is done by the control unit. Fig. 9 shows the operating flow between the program components.

The shader program declares several functions and variables to implement the color-thresholding method. First, the target color and all pixels in the retrieved frame will be normalized. Each of red, green, blue channel added and divided by three, then the whole color divided by that amount. The color vector is defined as

$$\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = \begin{pmatrix} r/n \\ g/n \\ b/n \end{pmatrix} \text{ where } n = \frac{r+g+b}{3} \qquad (1)$$

With the normalization step, we can attempt to remove the effect of varying lighting on colors. The next process is to calculate the color intensity level between the target color and every pixel in the image and define it as "distance":

$$distance = \sqrt{(r'-r)^2 + (g'-g)^2 + (b'-b)^2}. \qquad (2)$$

To quantize the frame to binary image, if the distance is within the threshold, we set the RGBA channel to 1.0, otherwise 0.0. When executing the decoding application, the shader program will update the declared parameters in the main program. Therefore, it is unnecessary for the user to reset the configuration, and these parameters will be updated in each frame.

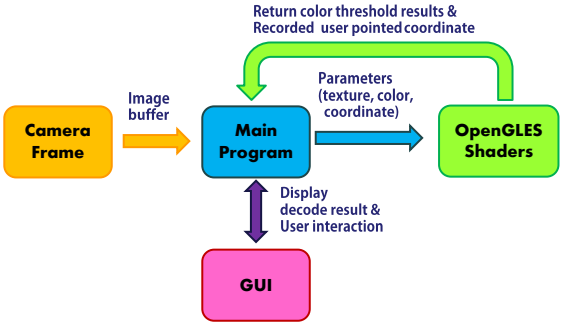


Fig. 9: Flow between program components.

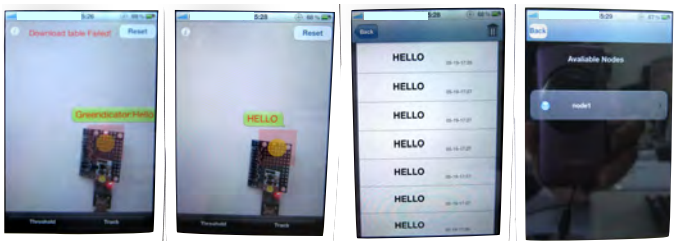


Fig. 10: Screen shots of decoding process with using iPhone 4S as decoder. (1) The welcome message. (2) After successful decoding. (3) The list of available nodes. (4) The recorded messages with timestamps.

C. Run-time Decoding

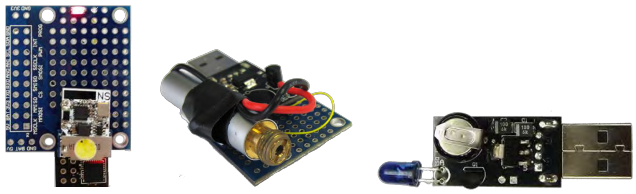
Fig. 10 shows the screen shot of the decoding process in Greendicator. Once started, if the decoder finds no available network, then it displays the alert messages "Table download failed!", which means the encoder should use ASCII encoding.

The red area on the screen is the detecting window, which contains the last user-pointed position. To run the color-threshold process, the user aims the detecting window at the target LED. Once the user selects the target color in the detecting window, the smartphone app displays a yellow dot on the screen to identify the tracked pixel and blinks the dot at the same rate as the LED.

After receiving the preamble, an instant-messaging dialog bubble pops up with a welcome message, and the smartphone starts the decode process and displays the decoded message. The user can also check the history of the messages by touching the information button on the main screen and checking the menu of available nodes. The recorded messages will be timestamped, enabling the user to obtain the complete status from the node.

VI. Evaluation

This section first describes our experimental setup, followed by our experimental results. Our metrics include the memory footprint, energy consumption, transmission throughput, and decoding robustness.



(a) (b) (c)

Fig. 11: (a) SMD LED (b) laser (c) IR transmitter.

TABLE I: Power consumption comparison through different encoding types and ways(Unit: mA)

	Min. power	Max. power
ASCII	3.16	4.01
Bi-directional ASCII	3.24	4.21
Bi-directional Table	3.20	4.19

A. Environmental Setup

1) Encoder and Decoder Platforms: We implement the Greendicator encoder subsystem on EcoSD, a resource-constrained wireless sensor platform equipped with an LED and the OP520 phototransistor. We implement the Greendicator decoder subsystem on an iPhone 4S and an iPod Touch 4. The difference between the two is that the iPhone 4S is equipped with an 8-megapixel camera and auto-focus functionality, whereas the iPod Touch 4 supports only "touch to focus" and with a lower camera frame rate. Nevertheless, the simplicity of the device can still help us experiment with our system in case the hardware support is limited. The flashlight enables bi-directional communication.

2) Optical Transmitter: We tested Greendicator with three different types of optical transmitters, including two visible-light ones and an infrared one. The first uses an SMD LED for short-range transmission and an OP520 phototransistor for reception. Both the LED and OP520 are mounted on the same expansion board for EcoSD and face the smartphone for bi-directional communication, as shown in Fig. 11a. The second uses a laser diode powered by USB on another expansion module as shown in Fig. 11b and is for long-distance transmission. The third uses the TSAL6400 DIP infrared-emitting diode and some logic circuits as shown in Fig. 11c. It is also powered by USB. For all transmitters, the EcoSD can use GPIO pins to control the diodes to send out the light pulses with accurate timing.

B. Experimental Results

1) Memory Footprint: Greendicator occupies 1932 bytes of code memory and 128 bytes of data memory. With bi-directional communication functionality, it costs 2482 bytes of code memory and 121 bytes of data memory. Both code and data memories consume only 10 to 13% of the total respective memories on EcoSD. The small memory consumption allows us to easily port Greendicator to other embedded platforms, whether constrained by memory or not.

2) Power Consumption and Energy Consumption: Greendicator reduces power consumption in several ways. Modulated LED pulses means lower power consumption by the LED than keeping it always on, while the additional power consumed by the MCU to perform modulation is trivial. Table I shows the power consumption of a sensor node with different encoding schemes. Both schemes consume less than 5 mA individually. Having bi-directional communication leads to higher power consumption than one-way, due to the fact that we use the timer and the OP520 phototransistor to periodically sense the light pulses from the smartphone, and these additional components increase the power consumption slightly.

Transmitter	Greendicator LED	Nordic 2.4 GHz RF
Current	4.1 mA	11.5 mA

Fig. 12: Power consumption comparison between different hardware components.

Transmitter	ASCII LED	Table LED	RF
Energy	372.9 mJ	14.619 mJ	0.33 mJ

Fig. 13: Energy consumption comparison with transmitting the same packet.

Fig. 12 depicts the current consumption of different hardware components when they are transmitting. Using a single LED results in the lowest current consumption of about 4 mA. However, the transmission rate for RF is much faster than modulated LED such that it can actually finish faster and goes back to sleep. We implement an experiment to further discuss the energy consumption.

In the experiment, we send the packet with the same content using ASCII LED, Table LED, and RF, separately. In the RF configuration, we set the transmission rate to 1 Mbps without auto-ack support. Fig. 13 shows the experimental results. The RF transmits about 1000 packets in six seconds, and the average transmitting current is 18 mA. ASCII LED needs to convey each character, thus costing 28 seconds to finish a packet, and the total energy consumption is the highest at 372.9 mJ. Table LED is much faster than ASCII LED since it only needs to convey the index of each message. It costs about one second to finish a packet transmission, which makes it a potential substitute for RF.

3) Transmission Throughput: The data transmission throughput when using Greendicator is limited by the camera frame rate. The average frame rate on the iPod Touch is 21 FPS, which is much lower than 29 FPS on the iPhone 4S. To ensure that the LED blinks can be fully captured even with limited hardware support, we set the minimum modulation period to 100 ms and the maximum to 500 ms.

Fig. 14 presents the difference of packet's payload length between ASCII and Table encoding schemes. We test blinking "HELLO" as a message followed by a delay of five seconds. As the result shows, the Table encoding scheme obviously shortens the payload length, since it needs to convey only the message index.

As debugging output for WSNs, Greendicator achieves an average throughput of 5.7 bits/s. If using iPhone as the decoder, then we can shorten the minimum modulation period to 70 ms and achieve a throughput of 1.2 byte/s. Although

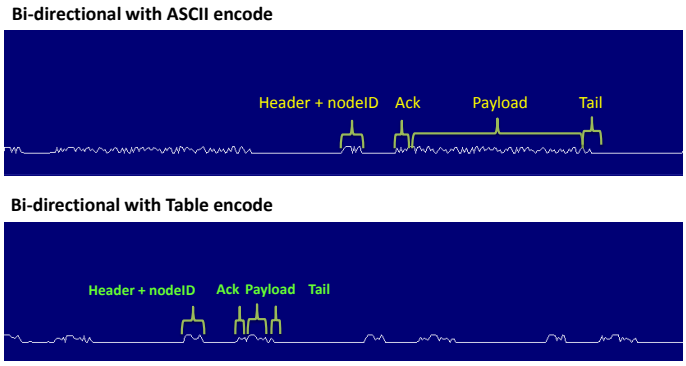


Fig. 14: Packet length with different encoding schemes

the throughput now cannot be compared to other wireless transmission technologies, we can use Table encoding method to blink a short index that represents a long string of messages to increase our throughput to about 2.1 seconds per message.

4) Decoding Robustness: We evaluate decoding robustness of Greendicator over different *distances*, *angles*, *encoding schemes*, and *degrees of optical visibility*. We also compare the decoding results between iPhone 4S and iPod Touch. The former is equipped with a powerful camera, while the latter represents the more constrained hardware platform.

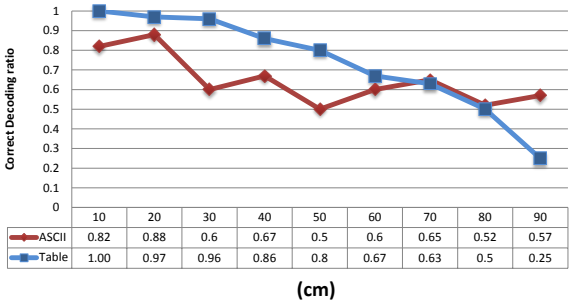
Short Distance with Different Encoding Schemes: We first discuss the relationship between decoding accuracy and distance over different encoding schemes. In the first experiment, we connect an EcoSD to an external LED module and secure it on the wall. The LED blinks the message "HELLO" in Table and ASCII encoding schemes, and we decode the message 20 times above at each distance. We define decoding accuracy as

$$\text{correct-decoding ratio} = \frac{\#\text{correctly decoded bytes}}{\text{total \#bytes decoded}} \quad (3)$$

We record the decoding results for the decoder distance from 10 cm to 90 cm in 10 cm increments.

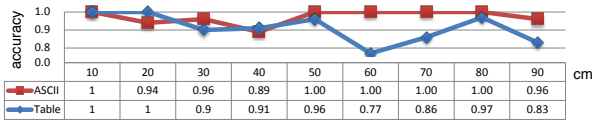
Fig. 15 depicts the results with using the iPod Touch as the decoder. As the results show, the correct-decoding ratio decreases in both encoding schemes when we move the decoder back away from the target LED to 90 cm. The decoding accuracy in Table encoding decreases more rapidly than ASCII encoding, possibly resulting from the fact that we do not apply the error-correction method in Table encoding. The average correct-decoding ratio in ASCII encoding is about 0.65. Although the performance is almost acceptable, we find that, in the most cases, at most one error occurs in the decoded message, which means the message can still be recognized by users.

Fig. 16 depicts the results of using iPhone 4S as the decoder. The overall performance is significantly better than using the iPod Touch. The average correct-decoding ratio is mostly above 90% by Table encoding and 97% by ASCII encoding. The higher correct-decoding ratio may be due to the higher-performance hardware. Since the iPhone 4S is equipped with an 8-megapixel camera and dual-core processor, it maintains the high resolution even when far from the target LED.



	10	20	30	40	50	60	70	80	90
ASCII	0.82	0.88	0.6	0.67	0.5	0.6	0.65	0.52	0.57
Table	1.00	0.97	0.96	0.86	0.8	0.67	0.63	0.5	0.25

Fig. 15: iPod Touch decoding accuracy at different distance.



	10	20	30	40	50	60	70	80	90
ASCII	1	0.94	0.96	0.89	1.00	1.00	1.00	1.00	0.96
Table	1	1	0.9	0.91	0.96	0.77	0.86	0.97	0.83

Fig. 16: iPhone 4S decoding accuracy at different distance.

The initial choice of the color threshold affects largely in the camera decoding result. If user defines the strict threshold, which means that higher similarity between pixels and the LED color is required, then fewer pixels will be interpreted as valid light pulses. Therefore, as the distance increases, the retrieved image of the LED becomes smaller and blurrier, increasing the difficulty for the decoder to capture the light pulses correctly. The iPhone 4S with auto-focus can strengthen the image quality, thus achieving a higher correct-decoding ratio, but the auto-focus feature may also cause instability in decoding.

Long Distance: In the second experiment, we apply a laser module as the optical emitter to transmit the message "HELLO" in ASCII encoding scheme. We do not directly focus on the laser but on the reflected point of light on the wall instead. We measure the correct-decoding ratio over several distances. The results are taken as the average of sending the same message 10 times at each distance. Fig. 17 shows that using the iPhone 4S as the decoder, within the range of 30 m, we achieve 100% correct-decoding ratio; even at the distance of 50 m, the correct-decoding ratio is still above 80%. This confirms the feasibility of using Greendicator in applications requiring long-range point-to-point communication.

Angle: In the third experiment, we find the relationship between the correct-decoding ratio and angles. We use the conventional LED and secure it on the wall. The decoder faces the LED from the horizontal position 0° to directly opposite position 90°. We tested several angles: 0°, 30°, 45°, 60°, and

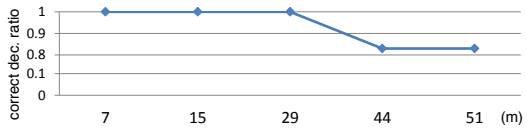


Fig. 17: iPhone 4S decoding accuracy at different distance through laser.

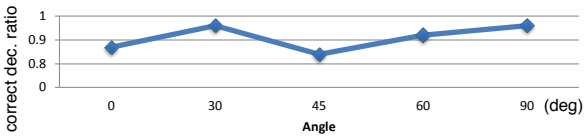


Fig. 18: iPhone 4S decoding accuracy over angles.

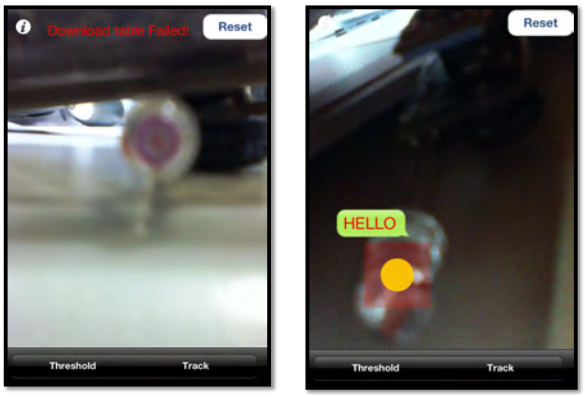


Fig. 19: iPod decoding IR-emitted message.

90° from the distance of 20 cm to the target LED, using the Table encoding scheme.

Fig. 18 depicts the results when using the iPhone 4S as the decoder. As the results show, the highest correct-decoding ratio occurs at angles 30° and 90°. Basically, at a near distance, the decoding angle does not impact the results much.

Optically Invisible Source: In the last experiment, we test the decoding performance of using an IR LED as the optical emitter. IR is optically invisible to human eyes, but it can be captured as a lilac color by the camera. The camera on other phones such as Nokia ones also exhibit similar behavior. Due to the non-obvious color, it is difficult to choose the correct color threshold, and the ambient light greatly affects the accuracy of camera detection. To filter out the environmental noise, we surround the IR emitter with a black cloth. Fig. 19 shows the detected IR pulses and the correct-decoding result. Although the IR emitter in our experiment can be decoded only within a very short range (about 10 cm), it presents the possibility of using optically invisible source as the message emitter.

VII. Conclusion and Future Work

We propose Greendicator, a method for enabling embedded systems to display text by encoding messages as LED pulses to be decoded by camera-equipped smartphones. This is particularly applicable to embedded systems that cannot have their own built-in displays (e.g., two-line LCD, bit-mapped LED or LCD, etc) for size reasons, as is the case with ultra-compact wireless sensor nodes. Greendicator enables virtually all embedded systems with an existing LED status indicator to convey both static and dynamic messages while occupying a small footprint of only 1932 bytes of code memory and 128 bytes of data memory in our implementation. One or more smartphones can all readily decode text from these blinking LEDs without RF setup. Moreover, Greendicator can work with different physical layers. The transmitter side can work with a visible LED, IR, laser diode, and even the reflected sunlight as the emitter, while on the receiver side, any camera-equipped device can be used as the decoder. Besides, Greendicator can be extended to bi-directional communication using the camera flashlight, making the system even more versatile.

As a message indicator, the average transmission throughput of Greendicator is 5.7 bit/s, which costs 1.4 s to transmit a raw byte. However, the throughput can be speeded up with Table encoding method, which costs only 2.1 s to transmit a complete message of arbitrary length. Greendicator provides an available decoding performance. With iPhone 4S, Greendicator maintains above 90% correct decoding ratio in the distance test and angle test. Thus, it can be a practical technology for many real-world applications. To ease programmers' burden, we provide API for programmer to apply Greendicator in their systems with a simple function calls.

We implemented Greendicator as a prototype. We currently use a single LED to convey messages, and the decoder tracks only one target node at a time. To achieve higher parallelism and increase the decoding efficiency, we can put multiple LEDs into an LED array and apply the hierarchical encoding method to convey different priority data at different distances [5]. This way, Greendicator can communicate with multiple target nodes at a time and at a higher data rate.

Also, since we apply simple color thresholding as the decoding method in Greendicator by OpenGL ES, to make our system more powerful, we can port the GPU-based computer vision library on iOS to enhance the image tracking functionality in our system. Several computer vision libraries are commonly used for image processing, including OpenCV, libCVD, and Cambridge Video Dynamics, etc. OpenCV is quite mature and covers most areas of computer vision, but now it runs only on CUDA platform, which is supported only on nVidia graphics cards.

Acknowledgments: This work is sponsored by Ministry of Economic Affairs (Taiwan) 100-EC-17-A-04-S1-044, National Science Council (Taiwan) NSC 101-2221-E-007-025.

References

[1] GPU-accelerated video processing on Mac and iOS. http://www.sunsetlakesoftware.com/2010/10/22/gpu-accelerated-video-processing-mac-and-ios.

[2] T. Hesselmann, N. Henze, and S. Boll. FlashLight: Optical communication between mobile phones and interactive tabletops. In *ACM Interactive Tabletops and Surfaces*, 2010.

[3] A. Mohan, G. Woo, S. Hiura, Q. Smithwick, and R. Raskar. Bokode: Imperceptible visual tags for camera based interaction from a distance. In *SIGGRAPH*, 2009.

[4] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. John Wiley & Sons, 2005.

[5] S. Nishimoto et al. Overlay coding for road-to-vehicle visible light communication using LED array and high-speed camera. In *Int'l IEEE Conf. on Intell. Transp. Sys. (ITSC)*, pages 1704–1709, Oct. 2011.

[6] H. Nyquist. Certain topics in telegraph transmission theory. *American Institute of Electrical Engineers, Transactions of the*, 47(2):617 –644, april 1928.

[7] J. G. Proakis. Digital communications, 3rd edition. 1995.

[8] D.-S. Shiu and J. Kahn. Differential pulse-position modulation for power-efficient optical communication. *Communications, IEEE Transactions on*, 47(8):1201 –1210, aug 1999.